www.elektor-magazine.com

elektor .magazine

December 2014 | No. 456

CoCo-ri-Co: Cool Controller Concept

Turn 'n Push Ultra-Versatile Human Interface for Microcontroller Systems

CoCo-ri-Co: The **Co**ol **Co**ntroller **Co**ncept | **T Board 28 Lowest-Power Exercising** | Programmable Christmas Tree | **VariLab 402 (2)** | ADS1115-eBoB | Arduino Software Development with Atmel Studio | Infinite RGB LED Cube with Flowcode | **Distance and Level Gauge** | **Improved Current Transformer** | **Red Pitaya** | DesignSpark Tips & Tricks | Tantalum Bead Capacitors | Bulging Caps | **Elektor.Labs goes wearable** | GestIC & 3D TouchPad Workbook (1) | GM2308 Audio Signal Generator (1950/64) | **Hexadoku**

Embedded 3D Gesture and Position Tracking

with Microchip's GestIC® Technology

Microchip's patented GestIC® 3D Technology provides a **single-chip**, real-time gesture recognition solution. The MGC3130 enables the next breakthrough in user interface design for any product at a very low cost point.

MGC3130

- The MGC3130 runs the Colibri Gesture Suite, enabling robust recognition of complex 3D gestures without requiring any host processing
- The MGC3130 delivers the results (detected gestures/ XYZ positioning) to the host via an I²C™ bus interface or using GPIOs

microchip DIRECT
www.microchipdirect.com

MICROCHIP

microchip.com/gestic

reichelt.com elektronik

Your competent online partner for

Components · Power supply systems · Measuring technology · Shop & soldering technology · Home & security technology · Network technology · PC technology · Sat/TV technology · Communication

SUBSCRIBE NOW!

Newsletter in English

Each week receive the latest information concerning
- New products
- Best offers
- Price reductions

STATE-OF-THE-ART SINGLE BOARD COMPUTERS

AT A GREAT PRICE!

Perfectly
matches the Banana Pi

TEKO

4-piece Cooling set
- Easy installation
- Extremely low profile

TEK-BERRY COOL (~ £ 3.65) € 4.65

WLAN USB adapter, 150 Mbit/s
EDIMAX
- Supports WEP, WPA, WPA2
- WPS compatible

EDIMAX EW-7811UN (~ £ 5.42) € 6.90

Micro-USB power supply 6W, 5V, 1,2A
- Double insulation
- Protected against overload and short circuit

HNP06 MICRO USB (~ £ 5.02) € 6.40

MicroSDHC card Class 10
Kingston Technology
- Including adapter
- Capacity: 8 GB

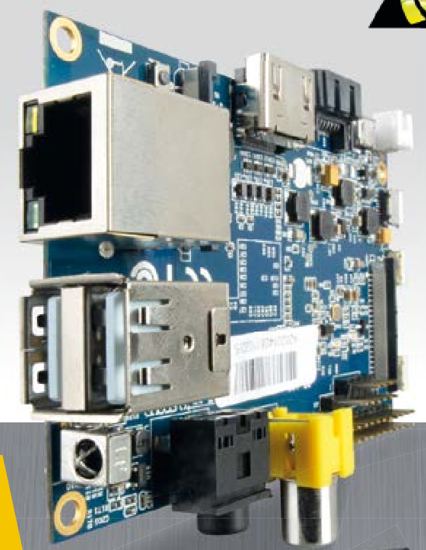
SDC10/8GB (~ £ 5.34) € 6.80

Banana Pi

Compact like the Raspberry Pi, but significantly more capacity!

- ☑ Faster CPU
- ☑ More RAM
- ☑ Fast Ethernet
- ☑ Infrared receiver

- CPU: A20 ARM® Cortex™-A7-Dual-Core
- RAM: 1 GB DDR3 SDRAM
- Connections: SD card slot, SATA port, HDMI, 10/100/1000 Mbit/s Ethernet, 2× USB port, CSI camera connection, 26 external PINs with I²C, SPI, UART, CAN, 3.5 mm audio output
- Power supply via micro-USB



BANANA PI €42.90 (~ £ 33.68)

Fits like a glove!
Plastic housing for your Banana Pi

CAMDENBOSS

Mix your own color combinations to suit your taste. Thanks to the clip-in grooves, bothersome screws or bonding are not required.

- All cut-outs are pre-cut
- Non-slip rubber feet
- High-gloss surface

€4.20

Yellow	Upper part CB DBBP AA GE	(~ £ 3.30)	
	Lower part CB DBBP BA GE	(~ £ 4.87)	€ 6.20
Transparent	Upper part CB DBBP AA TR	(~ £ 3.69)	€ 4.70
	Lower part CB DBBP BA TR	(~ £ 5.42)	€ 6.90
Black	Upper part CB DBBP AA SW	(~ £ 3.30)	€ 4.20
	Lower part CB DBBP BA SW	(~ £ 4.87)	€ 6.20
Carbon	Upper part CB DBBP AA CF	(~ £ 3.69)	€ 4.70
	Lower part CB DBBP BA CF	(~ £ 5.42)	€ 6.90

Delivery without Banana-Pi

Order now! **www.reichelt.com**

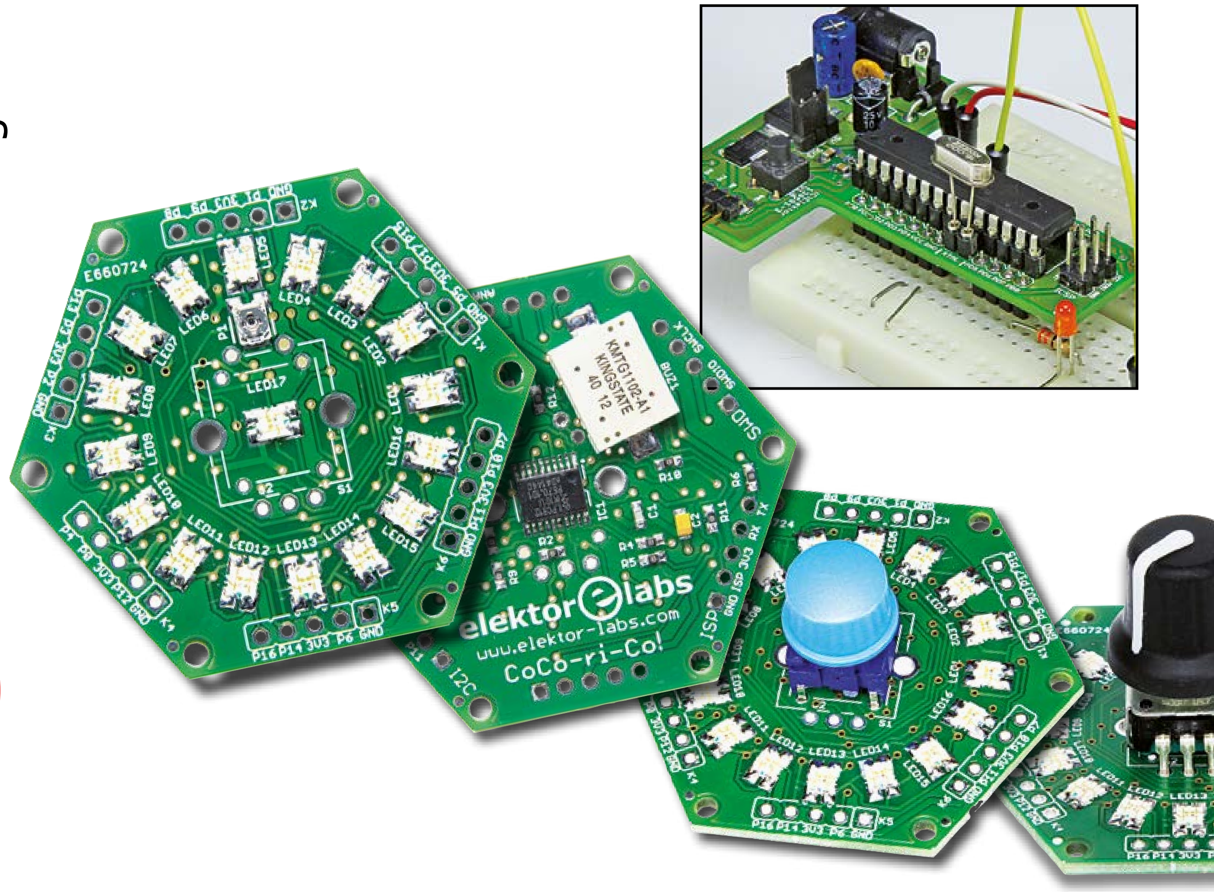
English Hotline: **+49 (0)4422 955-333**

Daily prices! Prices as of: 10/10/2014

Shop languages:

International payment method: PayPal

Contents



Projects

10 CoCo-ri-Co:
The **Co**ol **Co**ntroller **Co**ncept
A truly flexible, modular interface for microcontroller projects. Forget about switches, dials, displays, LCDs—with the Elektor CoCo-ri-Co drop-in human interface controller you just turn, push, and view the LED circle. It's clever, cheap, and available ready made.

20 T Board 28
Lowest-Power Exercising
Using the example of a standalone temperature logger we demonstrate how the power consumption of an Elektor T Board can be reduced drastically, to the level of "months on a single battery".

28 Programmable Christmas Tree
It's that time of the year again that we as electronics engineers provide some more unusual mood lighting for the home. Naturally Elektor will assist you again this year, in the form of a nice Christmas tree which conjures up all sorts of beautiful patterns on its 62 LEDs.

32 VariLab 402 (2)
This month we proceed with the control and display board for our high-end benchtop power supply. No worries, we have not forgotten USB for communication with a PC.

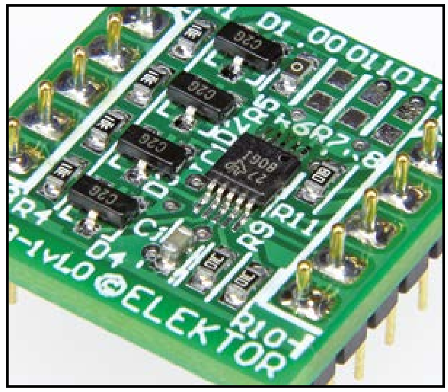
44 ADS1115-eBoB
The ADS1115 is a great device as far as ADCs are concerned but it's so small few of us have the means to actually solder the IC and use it in our projects. That's why Elektor made a breakout board (eBoB).

46 Arduino Software Development with Atmel Studio
Here we describe the glue between Atmel Studio, the Atmel SAM ICE and the Arduino Due. It makes debugging and programming a whole lot easier.

60 Infinite RGB LED Cube with Flowcode
Here's how to take a plain single-color 8x8x8 LED cube project to the Next Level. The newly released Flowcode 6 software development suite was instrumental in a lively journey into understanding and achieving the simulation of animations on 'full color' LED cube.

66 Distance and Level Gauge
Take a dirt cheap ultrasonic sensor module, add a display, a couple of pressbuttons and a microcontroller loaded with software, and you have all the ingredients for a lavish circuit doing range and distance measurement.

74 Improved Current Transformer
A method of improving on the results of an earlier publication.



Review

38 Red Pitaya
A technical look at the new test, measurement and signal processing platform with A/D and D/A, spectrum analysis, oscilloscoping—all running under Linux and open-source. We used it to do some RTTY generating and 10.7 MHz filter sweeping.

Labs

54 Bulging Caps
How a PC got rescued from the dumpster and resurrected in no time by replacing a few suspicious looking parts.

55 Elektor.Labs goes wearable
Spaghetti from the extruders and trinkets from the 3D Printer. More suggestions please, to Elektor.Labs.

DesignSpark

56 DesignSpark Tips & Tricks
On day 16 in our virtual month we learn about creating, editing, and updating components.

58 Tantalum Bead Capacitors
Weird Components—the series

Industry

8 GestIC & 3D TouchPad Workbook (1)
Here we connect the Microchip MGC3130 Gesture & Touch controller chip to the Raspberry Pi. The dev kit and TouchPad are on special offer at the Elektor Store.

76 News & New Products
A selection of news items received from the electronics industry, labs and organizations.

Regulars

80 Retronics: Two Philips GM2308 Audio Signal Generators (1950, 1964)
Mix the outputs of two 100-kHz oscillators, calibrate for zero with a magic eye, and use the beat frequency for audio testing and research. Courtesy Philips Physics Labs, and with tubes of course. Series Editor: Jan Buiting.

84 Hexadoku
The Original Elektorized Sudoku.

85 Gerard's Columns: Appearances Count
A column or two from our columnist Gerard Fonte.

90 Next Month in Elektor
A sneak preview of articles on the Elektor publication schedule.

Community

Volume 40, No. 456
December 2014

ISSN 1947-3753 (USA / Canada distribution)
ISSN 1757-0875 (UK / ROW distribution)
www.elektor.com

Elektor Magazine is published 10 times a year including double issues in January/February and July/August, concurrently by

Elektor International Media
111 Founders Plaza, Suite 300
East Hartford, CT 06108, USA
Phone: 1.860.289.0800
Fax: 1.860.461.0450

and

Elektor International Media
78 York Street
London W1H 1DP, UK
Phone: (+44) (0)20 7692 8344

Head Office:
Elektor International Media b.v.
PO Box 11
NL-6114-ZG Susteren
The Netherlands
Phone: (+31) 46 4389444
Fax: (+31) 46 4370161

USA / Canada Memberships:
Elektor USA
P.O. Box 462228
Escondido, CA 92046
Phone: 800-269-6301
E-mail: elektor@pcspublink.com
Internet: www.elektor.com/members

UK / ROW Memberships:
Please use London address
E-mail: service@elektor.com
Internet: www.elektor.com/member

Advertising & Sponsoring:
Johan Dijk
Phone: +31 6 15894245
E-mail: johan.dijk@eimworld.com

www.elektor.com/advertising
Advertising rates and terms available on request.

Happy 40th Anniversary Elektor

December 2014 marks 40 years of Elektor's English language edition. Browsing issue one dated December 1974 I found that Elektor founder Bob W. Van Der Horst's introductory words are to the point despite 40 years of revolutions in electronics land, including ICs, SMT, Microcontrollers and The Web. Below are a few paragraphs of Bob's 1974 introduction, the edits are mine.

>> This is the ~~first~~ fourhundred and fifty-sixth edition of Elektor, a magazine that introduces a new way of presenting electronics.
The Dutch edition of Elektor has been published for over ~~14~~ 53 years and the German for over ~~4~~ 43. Every month ~~120,000 copies~~ several terabytes find their way to ~~readers~~ members ranging from enthusiastic amateurs to professional electronic engineers.
Elektor's dynamic and practical application of new electronic techniques has stimulated the ever-present curiosity and imagination of engineers. Modern components, active and passive and especially cheap digital and linear circuits, are used in practical designs. Many of the circuits are developed in our own laboratories, and circuit building is greatly facilitated by using ready-made printed circuit boards we produce for the more important projects. […]
Elektor has always tried to be dynamic and informative: but it can occasionally irritate, as when it deflates technical imperiousness or indulges in humorous self-criticism that has given it a 'British' image on the continent.
We shall be working on the first copies for ~~1975~~ 2015 even as you read this. Articles already accepted describe an ~~electronically-compensated loudspeaker system~~ ARM controlled synthesizer, ~~a high-quality preamplifier~~ an AVR controlled function generator, ~~an analogue-digital converter~~ a LED replacement for 6-V moped lamps, ~~gyrators~~ Weird Components, and further developments of the ~~mos-clock~~ Platino, ~~electronic drum~~ T-Board and ~~TAP~~ ELPP. <<

Welcome back in 2014, Happy Reading,

Jan Buiting
Editor-in-Chief
Elektor International Media

The Team

Editor-in-Chief:	Jan Buiting
Publisher / President:	Don Akkermans
Membership Manager:	Raoul Morreau (all areas)
International Editorial Staff:	Harry Baggen, Jaime González-Arintero, Denis Meyer, Jens Nickel
Laboratory Staff:	Thijs Beckers, Ton Giesberts, Wisse Hettinga, Luc Lemmens, Mart Schroijen, Clemens Valens, Jan Visser, Patrick Wielders
Graphic Design & Prepress:	Giel Dols
Online Manager:	Daniëlle Mertens
Managing Director:	Don Akkermans

USA
Don Akkermans
+1 860-289-0800
d.akkermans@elektor.com

United Kingdom
Don Akkermans
+44 20 7692 8344
d.akkermans@elektor.com

Germany
Ferdinand te Walvaart
+49 241 88 909-17
f.tewalvaart@elektor.de

France
Denis Meyer
+31 46 4389435
d.meyer@elektor.fr

Netherlands
Ferdinand te Walvaart
+31 46 43 89 444
f.tewalvaart@elektor.nl

Spain
Jaime González-Arintero
+34 6 16 99 74 86
j.glez.arintero@elektor.es

Italy
Maurizio del Corso
+39 2.66504755
m.delcorso@inware.it

Sweden
Carlo van Nistelrooy
+31 46 43 89 418
c.vannistelrooy@elektor.com

Brazil
João Martins
+31 46 4389444
j.martins@elektor.com

Portugal
João Martins
+31 46 4389444
j.martins@elektor.com

India
Sunil D. Malekar
+91 9833168815
ts@elektor.in

Russia
Nataliya Melnikova
+7 (965) 395 33 36
Elektor.Russia@gmail.com

Turkey
Zeynep Köksal
+90 532 277 48 26
zkoksal@beti.com.tr

South Africa
Johan Dijk
+31 6 1589 4245
j.dijk@elektor.com

China
Cees Baay
+86 21 6445 2811
CeesBaay@gmail.com

Our Network

elektor
VOICE COIL
circuit cellar
hifi.nl MUZIEK - FILM - LIFESTYLE
audioxpress
techthefuture

Connects You To

Supporting Companies

Batronix
www.batronix.com/go/45 79

DLP Design
www.dlpdesign.com 19

Eurocircuits
www.elektorpcbservice.com 53

Express PCB
www.expresspcb.com 77

Front Panel Express
www.frontpanelexpress.com 19

India m2m
www.indiam2mforum.com 43

Labcenter
www.lacenter.com 92

Microchip
www.microchip.com/gestic 2

Pico
www.picotech.com/ps367 91

Pololu
www.pololu.com. 19

Reichelt
www.reichelt.com. 3

Saelig
www.saelig.com 51

Schaeffer AG
www.schaeffer-ag.de. 51

Not a supporting company yet?

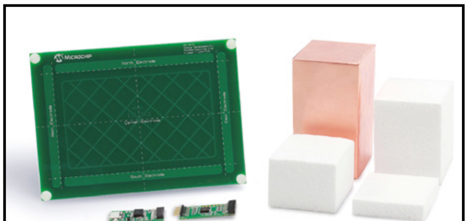
Contact Johan Dijk (johan.dijk@eimworld.com, Phone +31 615 894 245,
to reserve your own space in Elektor Magazine, Elektor•POST or Elektor.com

GestIC & 3D TouchPad WorkBook (1)

Raspberry Pi gets gesture control added

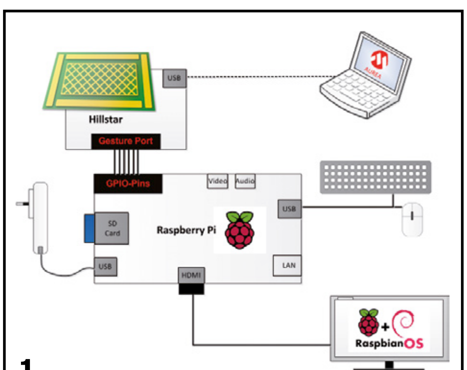
By **Thomas Lindner** & **Roland Aubauer** (Microchip GestIC® Team, Germany), and **Jan Buiting** (Elektor)

Last month we introduced the Microchip/Elektor 3D Gesture Control dev kit and 3D Touchpad product bundle. Here we showcase the potential of the kit by connecting the key component in the system, the MGC3130 GestIC® controller, to an RPi computer.



MGC3130 Hillstar Single Zone Development Kit (Part # DM160218)

Available in the Elektor Store



1

In response to introducing Microchip & Elektor's joint exclusive offering of a product bundle consisting of the Hillstar MGC3130 3D Gesture Control Development Kit ('dev kit') and a ready to use 3D Touchpad, the most frequently heard sighs and wishes were *I want to connect it to my micro!* Followed by *I want to connect it to my PC!* The offer was announced last month [1] and the hardware is on sale by the time this article appears in print [2].

In this short *WorkBook* series we'll step-demo the case of a dedicated, mixed touch & 3D touchfree, controller for connection to the Raspberry Pi computer, aiming eventually to play the '2048' game. It should enable the first wish to come true—more to follow.

This being a *WorkBook*, the idea is for You to get active with the technology involved, so we'll use up-tempo notes and tech scribbling. The information supplied here is intended to spur ideas for hardware and software development with the MGC3130 chip for 3D gesture control on any microcontroller system really.

1. What hardware

To be able to follow the project in an educational manner you need this hardware:

- Raspberry Pi Model B v.2, RBCA000 … 2x USB 2.0 3.5 watts
- Power supply: micro USB 1200 mA 5 V for RPi
- MGC3130 Hillstar Development Kit

The final sensor system works independent from the PC and just needs to be powered by a USB charger or directly from the RPi. Communication between the chips is established via the GesturePort pins provided on the little MGC3130 board in your Hillstar dev kit. A PC is required for parameterization and flashing the firmware to the MGC3130. A block diagram of the hardware configuration is shown in **Figure 1**, and a picture of the setup in **Figure 2**.

2. Connecting it up

The MGC3130 IC takes all the hassle out of capacitive sensing for 3D gesture control applications. The EIO1, EIO2, EIO3, EIO6, EIO7 signals on the Hillstar's MGC3130 board are connected to the RPi GPIO connector as pictured in **Figure 3**. Not forgetting Ground (GND) of course.

3. Parameterization with Aurea

The MGC3130 allows East-West and North-South hand flicks to be assigned to and flagged by its EIOx pins. **Aurea**, the free graphic shell around the GestIC controller, allows you to parameterize many sizes and configurations of planes that make up the capacitive sensing pad to be calibrated and configured with high precision. The parameters for the GesturePort are pictured in

Figure 4. Note the unique mix of 3D Gesture (Flick E ⇆ W; Flick N ⇆ S) and Touch (Center).

4. Programming stuff

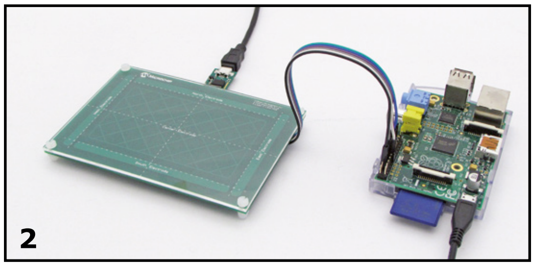
In terms of software, we will be using:

- OS: Raspbian (Debian Wheezy), version: January 2014, release date: 2014-01-07
- Python version: 2.7.3 (default, Mar 18 2014, 05:13:23), (already installed on Raspbian)
- RPi.GPIO library version: 0.5.4 (already installed on Raspbian)
- Tkinter (already installed on Raspbian)
- Leafpad text editor (already installed on Raspbian)
- The code for the game ‘2048_with_Gesture_Port_Demo.py’

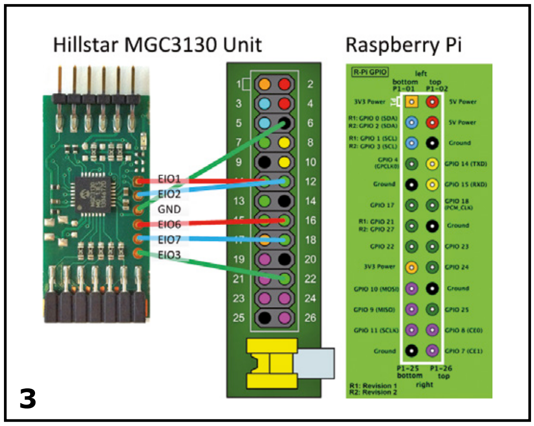
The game proper will be discussed and url’ed to you next month.

5. Quick Start

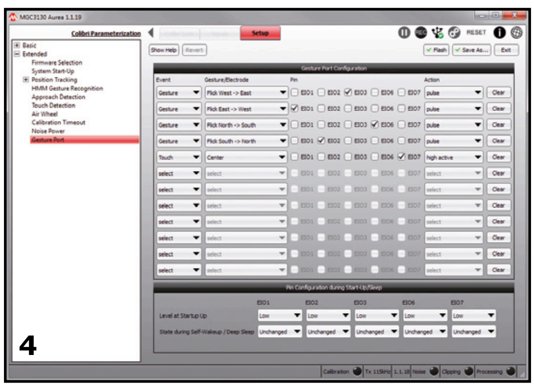
1. Install Raspbian on your Raspberry Pi.
2. Connect the Hillstar dev kit via USB to your PC and start Aurea. There should be a small popup window due to synchronization.
3. At the top side go to ‘Setup’ and then choose ‘Parameterization’.
4. Now click at the left side on ‘Extended’, browse in ‘Parameters’ for the pre-configured file ‘Hillstar GesturePort to Raspberry Pi Demo 2048.enz’ and start the parameterization, which takes a while.
5. After finishing, at the left side click on ‘GesturePort’ and you see the configuration for the events. If you click on ‘Flash’ in the upper right corner the configuration shown in Figure 4 is saved in the MGC3130 chip.
6. Once flashed, the Hillstar dev kit runs in standalone mode and you can connect the demo with your RPi as pictured in Figure 1. It can be powered either by an external USB charger via the mini USB connector or you steal 5 V from the Raspberry Pi.



2



3



4

Next month *GestIC & 3D TouchPad WorkBook* delves deeper in the demo and the *2048* game. Meanwhile Elektor Lab notes on the GestIC development kit and 3D Touchpad may also appear in the Elektor.POST newsletter.

(140423)

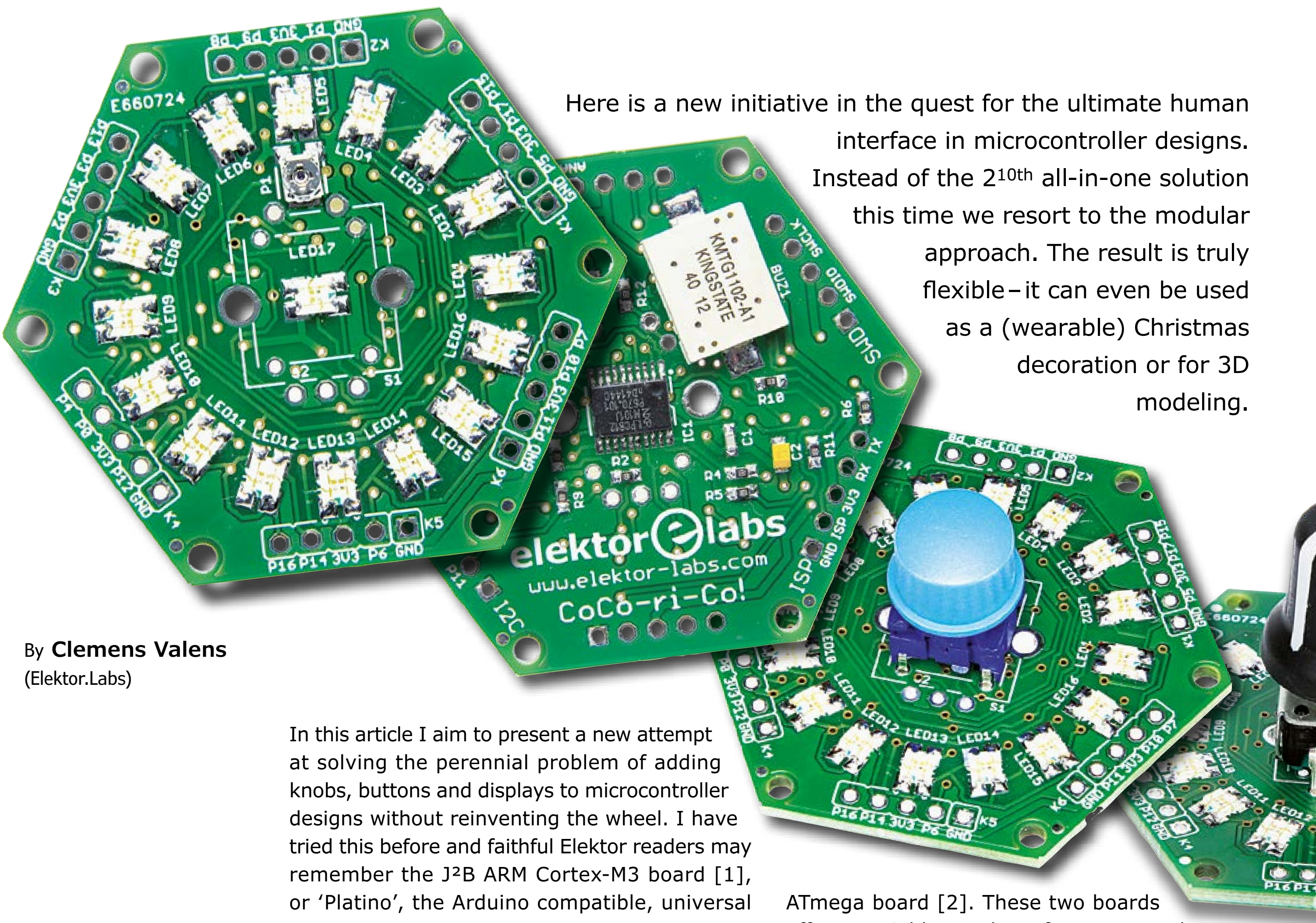
Web Links

[1] Add 3D Sensing to your Micro or PC. Elektor November 2014, www.elektor-magazine.com/140408.

[2] Microchip Hillstar GestIC dev kit and 3D Touchpad product bundle: www.elektor.com/microchip-dm160218-hillstar-development-kit-and-dm160225-3d-touchpad

[3] www.raspberrypi.org/downloads/

CoCo-ri-Co:
The Cool Controller Concept



Here is a new initiative in the quest for the ultimate human interface in microcontroller designs. Instead of the 2^{10th} all-in-one solution this time we resort to the modular approach. The result is truly flexible – it can even be used as a (wearable) Christmas decoration or for 3D modeling.

By **Clemens Valens**
(Elektor.Labs)

In this article I aim to present a new attempt at solving the perennial problem of adding knobs, buttons and displays to microcontroller designs without reinventing the wheel. I have tried this before and faithful Elektor readers may remember the J^{2}B ARM Cortex-M3 board [1], or 'Platino', the Arduino compatible, universal ATmega board [2]. These two boards offer a variable number of rotary encoders and pushbuttons together with a choice of LCD sizes, the idea being that one of these boards would satisfy the designer's appetite for controls.

This time I went for a different approach. Instead of many controls that can be positioned in different ways, this project only has one. Instead of one large versatile board that's supposed to contain everything, this design is for a small module that you sprinkle on or into your project like salt on an egg. No more bulky hard-to-read LCDs displaying cryptic messages in ugly fonts, this time we stick to LEDs. Bright lights visible

Specifications

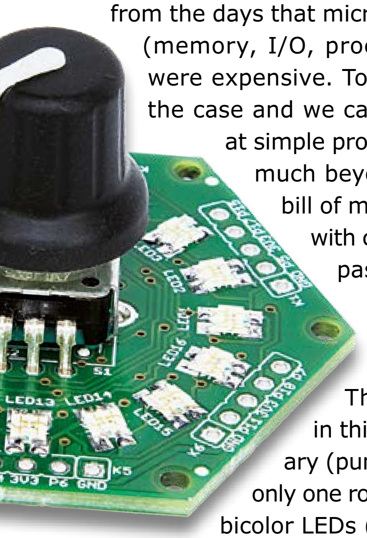
- NXP LPC812 32-bit ARM Cortex-M0+
- All 18 MCU I/O pins accessible through extension connectors
- Up to 17 bicolor LEDs
- Rotary encoder and/or pushbutton
- Buzzer
- Supports I^{2}C, SPI/synchronous and asynchronous serial communication
- ISP port compatible with 3.3-V FTDI USB-serial cable (except 5-V supply)

from a great distance and blinking fast, slowly or not at all is what we want; LCDs are uncool.

Compromises from the past

Many electronic devices are equipped with knobs and buttons that allow its user to adjust one or more parameters. Some of these controls do not need a precise indication of the value of the parameter they adjust. The volume control of an amplifier is an example. When the sound is loud (or soft) enough you simply stop turning the knob. Other controls—function selectors for instance—do need some indication to show their position but the resolution is not too important. A few LEDs are just fine in these cases. For controls that allow precise adjustments a numeric display is probably best.

Many applications use up/down pushbuttons to replace rotary controls even though the latter are faster and easier to operate. While it is true that pushbuttons are small, cheap and easily handled by microcontrollers, they remain a compromise from the days that microcontroller resources (memory, I/O, processing power, etc.) were expensive. Today this is no longer the case and we can throw lots of them at simple problems without adding much beyond one dollar to the bill of materials. So why stick with compromises from the past?



Reinvent the wheel

The design presented in this article is revolutionary (pun intended) in offering only one rotary encoder, up to 17 bicolor LEDs (i.e. 34 LEDs in total) and a buzzer controlled by nothing less than a 32-bit ARM Cortex-M0+ based microcontroller unit (MCU from now on). Overkill? Maybe, if you only look at processing power, but clearly no, also taking costs and interconnection potential into account. The MCU I chose for this project is the LPC812M101JDH20FP, LPC812 for short, featuring 18 I/O ports, 16 KB flash memory, 4 KB RAM, a good number of timers, an analog comparator, one I²C interface, two SPI controllers and three asynchronous serial communication interfaces. What's more, thanks to an integrated switch matrix, the pins of these communication peripherals can be connected to *any one* of the 18 available I/O ports. This cool feature means that it is possible to build a multi-protocol low-on-wires communications port that can be configured on the fly in software to talk either I²C, synchronous (SPI or similar) or asynchronous serial. Add a pin and full duplex communication becomes a reality. At a high-margin retailer's price of about 2 dollars apiece this clearly is a steal.

All components are mounted on a small circular printed circuit board (PCB) with the MCU, one LED and the rotary encoder in its center and the remaining 16 LEDs arranged in a circle around the encoder. At the edge of the PCB there is a ring of pinheaders for connecting the module to other modules to create a control surface or to wire it up to a host board.

Since there are only cool components on the board (ARM Cortex-M0+ MCU, bicolor LEDs, rotary encoder) it was baptized *Cool Controller Concept* or CoCo-ri-Co (the "ri" was added as a playful hint at the French way of writing the rooster's call). BTW, if you look hard at the board you will notice that it is hexagonal rather than circular. Round is cool, hexagonal is cooler; this is how CoCo-ri-Co reinvents the wheel.

Who needs this?

The main application of the CoCo-ri-Co board should be a digital potentiometer with instant position indication. Any device or appliance that could benefit from one or more rotary controls to adjust its parameters is a potential host to CoCo-ri-Co. Think of volume and tone controls in audio amplifiers, function selectors and speed controls on household appliances, temperature controls of heating systems or brightness and color controls of lamps—all of these applications make excellent hosts to one or more CoCo-ri-Cos. Because it can communicate using serial, I²C, SPI or custom protocols all through a single 3-pin port, CoCo-ri-Co can easily be connected to host applications that have a microcontroller inside. For those applications that do not have their own MCU, CoCo-ri-Co can be the main controller.

CoCo-ri-Co is small enough to fit in most enclosures, even in wall-mounted junction boxes and power outlets, and because of its hexagonal shape multiple boards can be tiled together to create a control surface. As an example, think of a speed

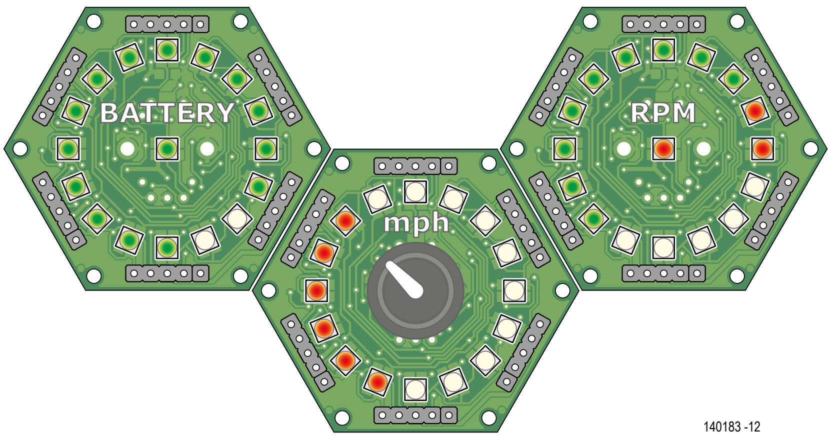


Figure 1.
With three CoCo-ri-Co boards you can construct a dashboard for an electric go-kart or another futuristic vehicle.

control with dashboard for an electric vehicle (**Figure 1**). One CoCo-ri-Co with encoder controls as a speedometer, another—encoderless—for displaying the battery level and a third might show the number of revolutions per minute (RPM) the motor is doing, or its temperature (or both). But there is more.

Thanks to its size and low-power requirements (oh, did I forget to mention that?) CoCo-ri-Co can be battery powered, from a 3 V button cell for instance, making it portable and even wearable (a geeky smart watch, anyone?). The bicolor LEDs allow for decorative applications and it excels at Christmas with its red and green looks and its buzzer happily beeping away at Christmas carols (see inset).

But that is not all.

Figure 2.
Get creative with polygonal PCBs. This soccer ball consists of 20 hexagonal CoCo-ri-Co boards and three pentagonal PCBs.

I already mentioned tiling boards together to create control surfaces, but these surfaces don't have to be flat. Indeed, you can create 3D surfaces from CoCo-ri-Co boards. The Cool Controller Concept may be extended to include pentagonal, square and triangular boards. With these shapes it is possible to create so-called Archimedean solids, you know, those highly symmetric, semi-regular convex polyhedrons composed of two or more types of regular polygons meeting in identical vertices. Like the stitched soccer ball.

Such a ball, mathematically known as a truncated icosahedron, is composed of 20 hexagons and 12 pentagons. A soccer or Bucky ball made out of CoCo-ri-Co boards (**Figure 2**) would be about 12 cms in diameter (25-mm vertex). Add a 3D accelerometer and it can show LED animations and play sounds depending on its position, speed and movement. Now how cool is that? CoCo-ri-Cool! I mean, totally cool.

What does it do?

Every time when the position of the rotary encoder changes CoCo-ri-Co will send a value (increasing or decreasing, depending on the direction of rotation) on the communications port (default: UART, 115200n81). This value lies between two boundaries (0 to 100 by default). The LED ring indicates the value as a bargraph (default) or a dot, in red (default) or green. The center LED can be controlled separately. Pressing the pushbutton will produce a Key-Down message and releasing it generates a Key-Up message. The messages can have the following formats:

ASCII mode

- Rotary encoder: E*xxxxx*, where *xxxxx* is a 5-digit field transporting values from "00000" to "65535".
- Pushbutton: B*x*, where *x* = '0' (button down) or '1' (button up).

Binary mode

- Rotary encoder: E*yz*, where the value is calculated as 256 *y* + *z*.
- Pushbutton: B*y*, where *y* = 0 (button down) or 1 (button up).

You can modify the behavior of the program on the fly by sending commands to CoCo-ri-Co. This allows you to change things like the color of the LEDs, the boundaries of the encoder value or the

Table 1. Commands to change parameters on-the-fly (* indicates default value). Every command requires two bytes.

Command	Data	Description
LEDs		
0x00 (0)	0|1*|2	LED ring color: off | red* | green
0x01 (1)	0*|1	LED ring mode: bargraph* | dot
0x02 (2)	0*-15	LED ring first LED (0*)
0x03 (3)	0-15*	LED ring last LED (15*)
0x04 (4)	0*|1	LED ring direction: clockwise* | anti-clockwise
0x05 (5)	0*-15	LED ring rotation (0*)
0x06 (6)	0-15	LED ring LEDx on
0x07 (7)	0-15	LED ring LEDx off
0x08 (8)	0|1*|2	Centre LED: off | red* | green
Rotary Encoder		
0x10 (16)	0*|1	Encoder mode: normal* | accelerated
0x11 (17)	0-255 (0)	Encoder minimum, low byte (0*)
0x12 (18)	0-255 (0)	Encoder minimum, high byte (0*)
0x13 (19)	0-255 (64)	Encoder maximum, low byte (64*)
0x14 (20)	0-255 (0)	Encoder maximum, high byte (0*)
0x15 (21)	0-255 (0)	Encoder value, low byte (0*)
0x16 (22)	0-255 (0)	Encoder value, high byte (0*)
Buzzer		
0x20 (32)	0|1*	Buzzer disabled | enabled*
0x21 (33)	1	Buzzer beep now
0x22 (34)	0-255 (232)	Buzzer beep frequency [Hz], low byte (232*)
0x23 (35)	0-255 (3)	Buzzer beep frequency [Hz], high byte (3*)
0x24 (36)	0-55 (64)	Buzzer beep duration [ms], low byte (64*)
0x25 (37)	0-255 (0)	Buzzer beep duration [ms], high byte (0*)
Other		
0xfe (254)	0*|1	Output mode: ASCII | binary
0xff (255)	1	Restore default values

encoder value itself, the mode of the encoder, the range of LEDs to use (default is 0 to 15). You can also rotate the bargraph, produce a beep, etc.

At the time of writing this article I am still adding useful commands, so please refer to the free downloads for the complete list. Updates for this project will be posted on Elektor.Labs [3]. **Table 1** lists the currently implemented commands.

Configuration

One-time configuration of CoCo-ri-Co is by means of trimpot P1. Cofiguration consists mainly of choosing the communications port type to use (UART, SPI/synchronous or I²C). With P1 you can select up to 32 values in theory, but for practical reasons—the precision of P1 being the most important—this has been limited to eight ranges. In **Table 2** you can find which functions

Table 2. Configurations selectable with P1. In SPI/synchronous mode, make sure that when powering on the system the SCK line (P12) is logic High (3.3 V), otherwise CoCo-ri-Co will boot into ISP mode.

Range	Configuration	Note	Signals
1	UART, no parity, 8 data bits, 1 stop bit (n81)	baudrate = 115200	P4=TxD, P0=RxD
2		baudrate = 38400	
3		baudrate = 9600	
4	SPI/synchronous, Slave		P4=MISO, P0=MOSI, P12=SCK (SSEL=?)
5			
6	I²C, Slave	address 0x1b (29)	P4=SDA, P0=SCK (P12=IRQ)
7		address 0x3a (58)	
8		address 0x74 (116)	

are activated by a range.

The ranges are defined such that turning P1 to its minimum value selects UART mode, setting it in the middle gives SPI/synchronous and at its maximum CoCo-ri-Co will talk I²C. A more precise procedure is available also and goes like this:

1. Switch off the board.
2. Fit a jumper between P6 and GND (connector K5, pins 1 and 2).
3. Switch on the board.
4. Adjust P1 to make the LED ring light the range you want. Stay in the middle of a range, i.e. away from range boundaries or you may experience random mode changes in the (near) future.
5. Switch off the board.
6. Remove the jumper between P6 and GND (connector K5, pins 1 and 2).
7. Switch on the board and test.

Do not fiddle too much with P1 because it is a tiny and fragile device.

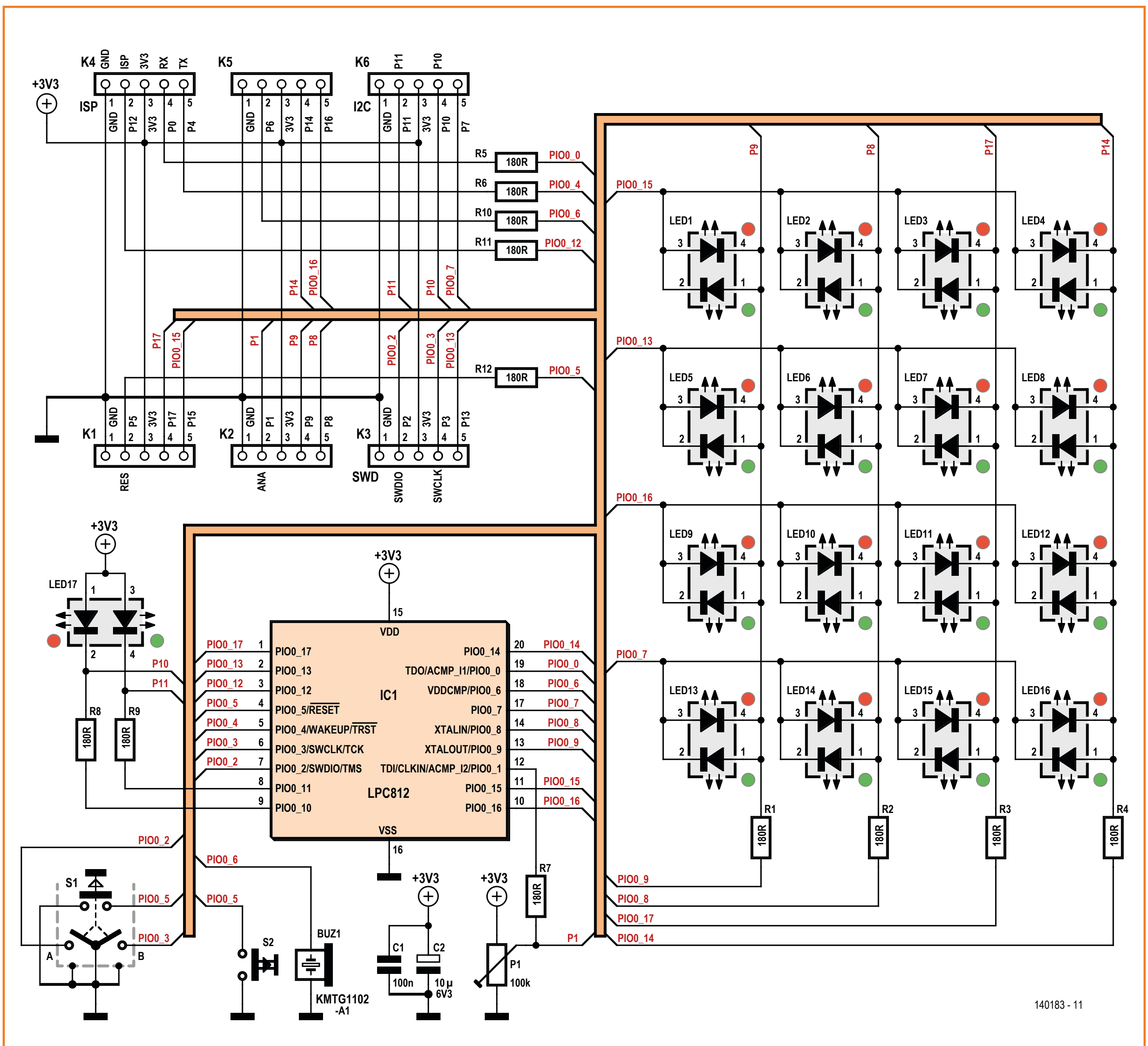


Figure 3.
Schematic of CoCo-ri-Co. The LPC812 ARM Cortex-M0+ microcontroller has more than enough brains and muscle to take care of everything.

Note that the SPI/synchronous and I²C Slave modes require the host application to poll CoCo-ri-Co on a regular basis. You could use a third port to interrupt the host when new data is available. Also note that real SPI Slave mode using the MCU's SPI peripheral requires the SSEL signal. It is up to you to decide which port to use for this.

The "SPI" implemented uses the USART in synchronous mode that can work without a SSEL signal. If Slave mode doesn't suit you, be aware that Master mode is available too (after some programming).

Into the schematic

After introducing the board, its functions and capabilities let's have a look at its schematics in **Figure 3**. Connecting 34 LEDs (bicolor, remember?) directly to 18 I/O ports is hard if not impossible, but when connected as two anti-parallel 4 x 4 matrices (16 LEDs per matrix) they only eat up 8 pins of our budget. These 16 LEDs form the circle. This leaves one bicolor LED—positioned in the center of the board—that must be connected in another way. Little known, the LPC812 MCU I²C module supports two types of interfaces: 'normal' capable of using any I/O pin, and 'dedicated' insisting on using ports 10 and 11. These two ports have high current sink capabilities and are therefore great for controlling LEDs, so I wired the remaining bicolor LED to these ports. We will content ourselves with normal I²C for communication.

To limit the current through the LEDs resistors are connected in series with them. Normally the red and green LEDs would require different valued resistors to obtain the same perceived brightness (red usually being brighter than green) but for the LED matrix this is not possible. It would have been possible to do this for LED17, but for convenience's sake I only used one value resistor on the board: 180 Ω.

For the rotary encoder I picked a model with integrated pushbutton. Encoders without pushbutton are about five times cheaper, but I like the turn 'n' push way of setting parameters. The pushbutton is hooked up to the MCU's Reset pin. That may seem strange, but it makes sense once you know you can disable the reset function on this pin in software. During software development you would normally keep the reset function available, but once the application is finished you can remove it. Or mount an encoder without an integrated pushbutton. For those applications requiring a pushbutton only, the PCB has a special pushbutton footprint too, wired in parallel with the encoder's pushbutton.

The encoder is in the center of the board, on top of LED17. Why? First, your application may need only one of the two components, so you mount only what you need. Second, we can have shaft encoder lighting. LED17 allows you to use the encoder shaft as a light guide (of course, you will need an encoder with a transparent shaft for this). The pushbutton footprint is also in the center of the board but as far as I know the model I chose does not come in a transparent package. In any case, LED17 can produce a sort of cool background glow of the control.

Many devices beep, or, as serious engineers would say, they provide audible feedback. A module presented as a universal control module must have the possibility to do the same. Therefore a footprint for a buzzer wasn't forgotten. Due to space constraints I had to use an SMD buzzer, but unfortunately they can be pretty expensive.

A variable resistor P1 is attached to port PIO0_1, which is connected inside the MCU to an analog comparator. This comparator can distinguish between 32 levels that make it useful, for example, as a 32-position switch to select different software functions. It is mounted at the LED side of the board for easy access (though it's pretty small). Do not mount P1 if you have no need for it.

The MCU does not require an external crystal or oscillator because it has a good one built in. If you need a better clock you should wire it up to PIO0_8 and PIO0_9 but the board is not equipped with footprints to make this an easy job.

All resistors on the board have the same purpose (current limiting) and value (180 Ω). They protect most of the 18 I/O ports, but not all. The two SWD pins PIO0_2 and PIO0_3 are unprotected so as not to interfere with any debugging pods. The LED matrix row pins are not protected either because it is unlikely for them to see any use outside the board, and board space is limited. Capacitors C1 and C2 provide power supply filtering.

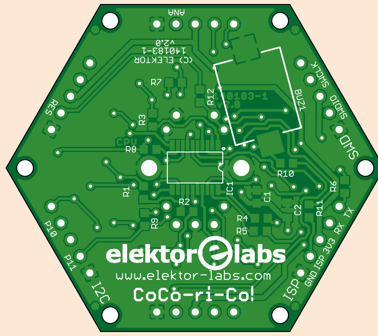
Component List

Resistors
R1–R12 = 180Ω, SMD 0603
P1 = 100kΩ trimpot, 3mm, e.g. Bourns TC33X-2-104E

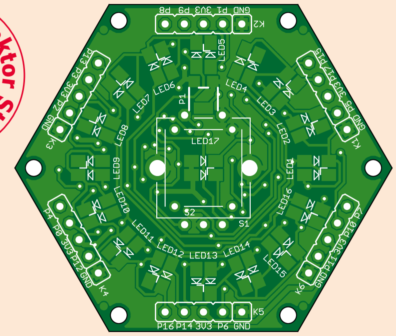
Capacitors
C1 = 100nF, SMD 0603
C2 = 10µF 6.3V, SMD 0805

Semiconductors
IC1 = LPC812M101JDH20FP (TSSOP20)
LED1–LED17 = LED, bicolor red/green, Dialight 5988610207F

Miscellaneous
BUZ1 = buzzer, Kingbright KMTG1102-A1
S2 = pushbutton, Multimec 3FTL6



Available in the Elektor Store



S1 = rotary encoder with/without integrated pushbutton, e.g. Alps EC12E2424407
K1–K6 = 5-pin pinheader, horizontal or vertical (optional)

Ready assembled module: Elektor Store # 140183-91, see www.elektor.com
Optional: PCB only, Elektor Store # 140183-1 (v2.0)

Extension connectors

The MCU also is located in the center of the board, but at the solder side with the resistors, capacitors and buzzer. All of its 18 I/O pins have been routed to six 5-pin extension connectors, each giving access to three I/O ports (6 x 3 = 18) plus 3.3 V and GND. These connectors are not wired up in a random way, but so that they have a useful function. K1 and K4 for instance provide in-system programing (ISP) capabilities. K4 is the real ISP connector (almost!) compatible with a 3.3 V FTDI USB-to-serial converter cable and K1 adds the Reset pin to it. They are on opposite edges of the board, 1.5 inch apart, so it is easy to construct a more elaborate programming adapter on prototyping board if you would like to. Such an adapter would probably have a 3.3-V regulator on it to transform the 5 V from the 3.3-V FTDI cable (yes, beware!) to 3.3 V. Actually, the Elektor 110553 BoB-FT232R USB-to-serial adapter [4] is preferable to the FTDI cable because it can provide the 3.3 V too.

K2 is considered to be an analog port, because it is connected to the MCU's analog comparator's input 2 (–––––ACMP_I2, PIO0_0 can be used as input 1). Together with PIO0_8 or PIO0_9 you can use it to create for instance a capacitive touch sensor. If P1 is mounted you will have a variable voltage on pin 2 of this connector.

K3 provides access to the serial wire debug (SWD) port of the MCU. It shares this port with the rotary encoder but when in rest its contacts should be open (if you use a model with detents and it is positioned between the detents). The pinout of K3 is not conforming to any SWD pod standard that I know of so you will probably need an adapter of some sort.

K5 is intended for digital I/O. Its pin 2 (PIO0_6) has a current limiting resistor because this pin is not 5V tolerant (all the others are). The other current limiting resistors are on ports that may be frequently connected and disconnected.
K6 is the full-speed I²C interface, connected to the pins 10 and 11 (and the center LED).

Of course most ports on the extension connectors are shared with the LED matrix and the other

Table 3. I/O ports in relation to extension connectors and functions.

PIO	K	Function	PIO	K	Function	PIO	K	Function
0	4	RxD/Free	6	5	Buzzer	12	4	ISP/Free
1	2	P1/Analog	7	6	Row 3	13	3	Row 1
2	3	SWDIO/S1A	8	2	Column 1	14	5	Column 3
3	3	SWCLK/S1B	9	2	Column 0	15	1	Row 0
4	4	TxD/Free	10	6	LED17R/I²C	16	5	Row 2
5	1	Reset/Pushbutton	11	6	LED17G/I²C	17	1	Column 2

on-board peripherals so beware when you start wiring the extension ports to all kinds of other components. **Table 3** summarizes the relations between the I/O ports, connectors and functions. From this table it is clear that connector K4 is meant for communication with the host application. In ISP mode (i.e. when the MCU is reset while PIO0_12 is being held Low) PIO0_0 and PIO0_4 become a serial port. In normal operating mode these pins are free and can be assigned to for instance the SPI or I²C module or to something else altogether. Once out of reset PIO0_12 becomes a normal I/O port. Note that PIO0_12 does not need a pull-up resistor because it already has one inside the MCU (as do all the I/O ports, except PIO0_10 and PIO0_11) so ISP mode will not be activated accidentally when this pin is left unconnected.

The project software

NXP has a website dedicated to the LPC microcontroller products [5] where you can find libraries for all of them and which saves you the work of creating the low-level hardware abstraction layer. These libraries come in pairs, one for the chip family itself and another for the evaluation and/or demonstration board. Since this project uses the LPC812 we need to look in the LPC8xx packages, and since we use the LPCXpresso compiler tools (**Figure 4**) we have to pick the package for this tool. The libraries are included (version 2.01) in the download for this article [6]. I urge to not replace them with freshly downloaded versions without doing an extensive file compare first, because I have corrected some bugs in my libraries that I have come across.

The LPCXpresso workspace for CoCo-ri-Co contains two projects: `lpc_chip_8xx_lib` (the NXP

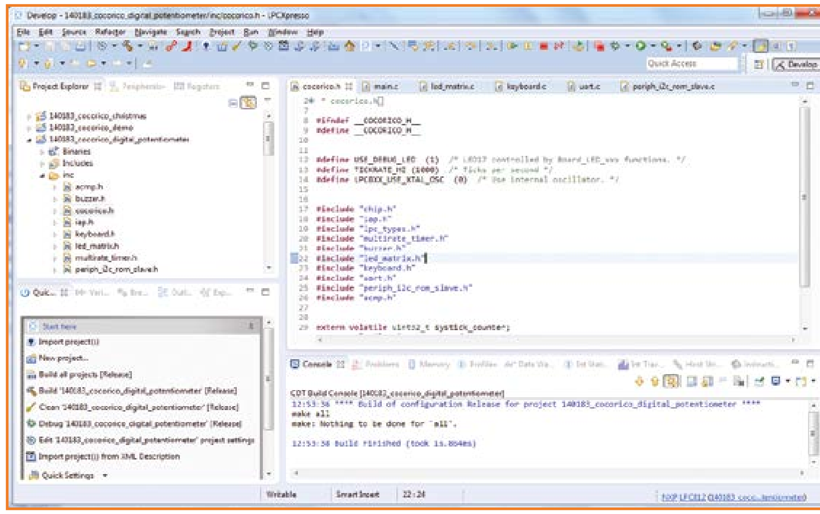


Figure 4.
The LPCXpresso IDE used to develop the CoCo-ri-Co software. Although you cannot see it in this screenshot, the project named lpc_chip_8xx_lib is also part of the workspace.

Sing-along Christmas Karaoke

CoCo-ri-Co unites 17 bicolor (red/green) LEDs and a buzzer on a small, green, roundish printed circuit board suitable for powering from a 3-V button cell. It has holes in every corner. What more do you need to create fun Christmas objects? Hang it in the Christmas tree, wear it as a broche, as an earring, on your wrist or wrap one up—it is the perfect gift.

To help you getting started I created a special CoCo-ri-Co Christmas project that plays Jingle Bells accompanied by its joyful blinking LEDs. Using a suitable adapter, connect the serial port to a PC or laptop and use a terminal program like Tera Term to display the lyrics Karaoke-style. Invite your beloved ones around the computer screen (or TV, or project it on a wall) and sing along with CoCo-ri-Co. Cozy atmosphere guaranteed.

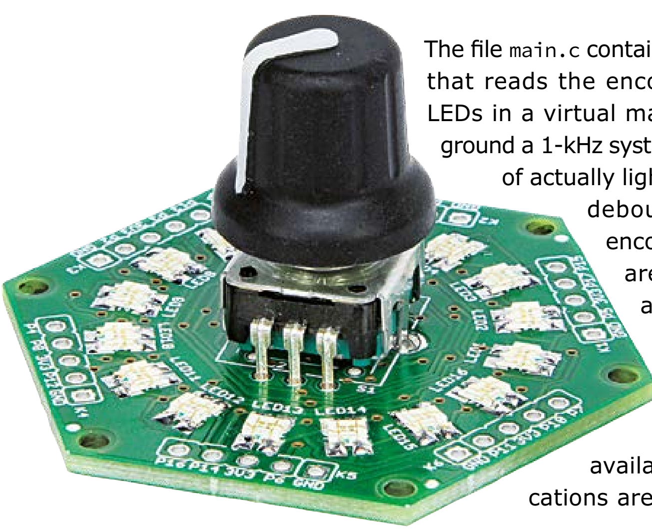
The beat is indicated by the center LED and can be adjusted with the rotary encoder (mount it on the back so the center LED remains visible).

See [3] and/or [6] for the files. The project is set up in such a way that it is easy to modify the song and the lyrics. Christmas will never be the same again without this brilliant gadget.

Things to know about LPC812 I/O ports

- In Table 3 the ports 10 and 11 are listed as I²C without specifying a signal (SDA or SCK). That's because it is up to you to decide which port carries what signal. See the Switch Matrix section in the LPC81x User manual. These ports have open-drain outputs and should not be used for fast SPI or UART modes.
- Ports 2, 3 and 5 default to their SWD and Reset function on power-up and must be reconfigured in software before they can be used for other purposes. Pin Enable Register 0 controls these pins.
- In digital mode all ports are 5-V tolerant except for PIO0_6. Ports that can have analog functions (PIO0_0, PIO0_1 and PIO0_6) become ***5-V intolerant*** when their analog function is activated.
- If the software disables the Reset input it is still possible to enter ISP mode simply by holding PIO0_12 low while power-cycling the MCU (off-on).
- The LPC81x family comes in several packages. On early versions of the 16- and 20-pin packages the ISP mode entry pin was PIO0_1 instead of PIO0_12. If your MCU refuses to go into ISP mode, check the revision code. It must be "4C" or higher for PIO0_12 to work.
- A virgin LPC81x MCU boots directly into ISP mode so you should be able to program the MCU at least once.

library) and `140183_cocorico_digital_potentiometer` (the real stuff). Both are included in the downloadable archive that you can import in LPCXpresso without unpacking it first.



The file `main.c` contains the main engine that reads the encoder and sets the LEDs in a virtual matrix. In the background a 1-kHz systick timer takes care of actually lighting the LEDs and debouncing the rotary encoder. Buzzer beeps are produced by yet another timer (the Multi-Rate Timer). To save program memory and because they are available the communications are handled as much as possible by the MCU's built-in drivers.

I encourage you to have a look at the source code (in plain C) and play around with it. Copy the digital potentiometer project to a new name in the same workspace and modify it. It is not too difficult I hope and comments are plentiful. When you build a project it will create a HEX file that you can program into the MCU using a serial port and the Flash Magic tool [7].

CoCo-ri-Co is available as a ready assembled module from the Elektor Store. Bare PCBs are also supplied for those who insist on an all-DIY build at home or in the lab. Questions and contributions can be posted on the Elektor.Labs page of the project [3].

(140183-I)

Web Links

[1] J²B: Elektor September 2011, www.elektor-magazine.com/110274; www.elektor-labs.com/node/3832

[2] Platino: Elektor October 2011, www.elektor-magazine.com/100892; www.elektor-labs.com/node/2288

[3] CoCo-ri-Co on Elektor.Labs: www.elektor-labs.com/node/4257

[4] Elektor 110553 BoB-FT232R: www.elektor.com/110553

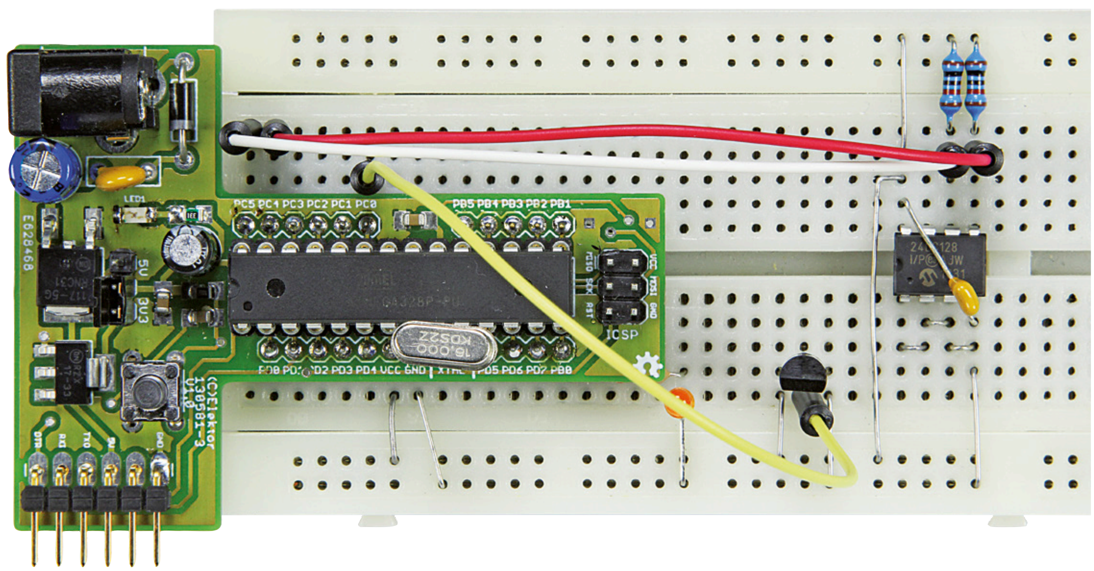
[5] LPCOpen: www.lpcware.com

[6] Project downloads: www.elektor.com/140183

[7] Flash Magic programming tool: www.flashmagictool.com

T-Board 28 Lowest-Power Exercising

Temperature Logger beats Arduino hands down



By **Andrew Retallack** (South Africa)

With the Internet of Things increasingly driving MCU-based projects, power consumption is becoming a critical element of any design. Every milliamp counts when remote sensors need to run for months or years on a single battery. Elektor's latest T-Board 28 is perfectly placed to help you prototype your next low-power project, as illustrated in this build of a power-efficient standalone temperature logger. Put on your t-shirt!

The T-Board 28 [1] has a number of features that give a designer the ability to optimize the power utilization by tweaking the design and then measuring the impact of these. This is one of the key benefits to the T-Board over other fixed platforms such as the Arduino. As many first-time users of the T-Board will have made the jump from an Arduino platform, I thought it would be relevant to show my own experimental approach using the T-Board 28 to progressively reduce the power consumption of a temperature logger. The T-Board 28 comes with an ATMEGA328 micro fitted, and is the largest of the T-Boards series. The series and the t-shirt are in the Elektor Store.

Why a temperature logger?

In order to become more familiar with ways of reducing power consumption, I wanted to work on a relatively simple project that would have a practical use without distracting me from my power-optimization goal. A temperature logger was a perfect fit. The project was built to measure temperature at regular intervals using a

low-cost temperature sensor, recording these on an EEPROM chip. When connected to a PC (or a laptop in the field), the readings can be downloaded over a serial connection. SD Cards were considered in the initial design, but with significantly higher power consumption (up to 3.5 times) and lack of tolerance for 5-V supply, it was decided to use EEPROM instead. At a later stage the project could be enhanced to include RF transmission of the readings; power-optimization of RF is of course a whole other ball-game!

A quick look at the project design

As can be seen from the schematic in **Figure 1**, the design is very simple (a major advantage of using a T-Board). A Microchip 24LC128P EEPROM was connected via I^{2}C to a T-Board 28, with the required pullup resistors on the I^{2}C communication lines. An LM60 temperature sensor was selected due to its low-current drain, ability to easily record negative temperatures (for those readers and editors not lucky enough to live in sunny South Africa!), and importantly its support for voltage inputs starting at 2.7 V. Finally, an LED was connected to give a visual indication of the tasks the logger was performing. This helped when measuring the power consumption, and would of course be removed in the final version of the logger.

The 24LC128, as with all I^{2}C devices, has a device address. This address can be set by tying pins 1, 2, 3 (A0, A1, A2) either High or Low. The 7-bit address of the device starts with a fixed ‘1010’ followed by the logic levels on pins A2 to A0. For this design (with A0 tied High) the address is therefore 1010001. When you wire the chip onto the breadboard, pay careful attention to all pins—I spent hours debugging my code, only to find I had tied pin 7 (the Write Protect pin) to V_{CC} instead of V_{SS} (GND) effectively write-protecting the chip! Datasheets are easily misread.

Serial I/O (for debugging and producing a dump of stored readings) is carried out over the FTDI pins of the T-Board, using the ATmega328’s built-in UART. An FTDI cable or module, connected to the T-Board’s FTDI connector, allows the T-Board to be connected to a PC’s USB port. Further iterations of the design could include porting the code to the T-Board 8, which would require a software-implementation of the Serial protocol.

The program code

The code was written using Atmel Studio 6.2, the latest version of Atmel’s in-house IDE. Simple modules were written to handle the ADC (Analog-to-Digital Converter), TWI (Two-Wire Interface), and UART functions. Here we need to digress briefly to discuss terminology. Due to trademarking around the I^{2}C protocol, Atmel have called their I^{2}C-compatible protocol TWI. It is 100% compatible with I^{2}C, but is just not called I^{2}C. Therefore in the datasheets and register naming you’ll be looking for TWI, not I^{2}C.

The code is available for downloading at [7] It should be straightforward to follow, however a few notes may be needed for the compiler symbols (accessed through the *Symbols* option of the *AVR/GNU C Compiler* section of the project properties). In order to calculate communication speeds and interval timing accurately, `F_CPU` defines the CPU speed in hertz. System voltage is defined in millivolts by `SYSTEM_MILLIVOLTAGE`, to allow ADC readings to be correctly converted into voltages. A `DEBUG` symbol exists that will output debugging information on the serial port if set to ‘1’.

When the MCU powers on, it first displays a simple menu over the Serial port to allow the user

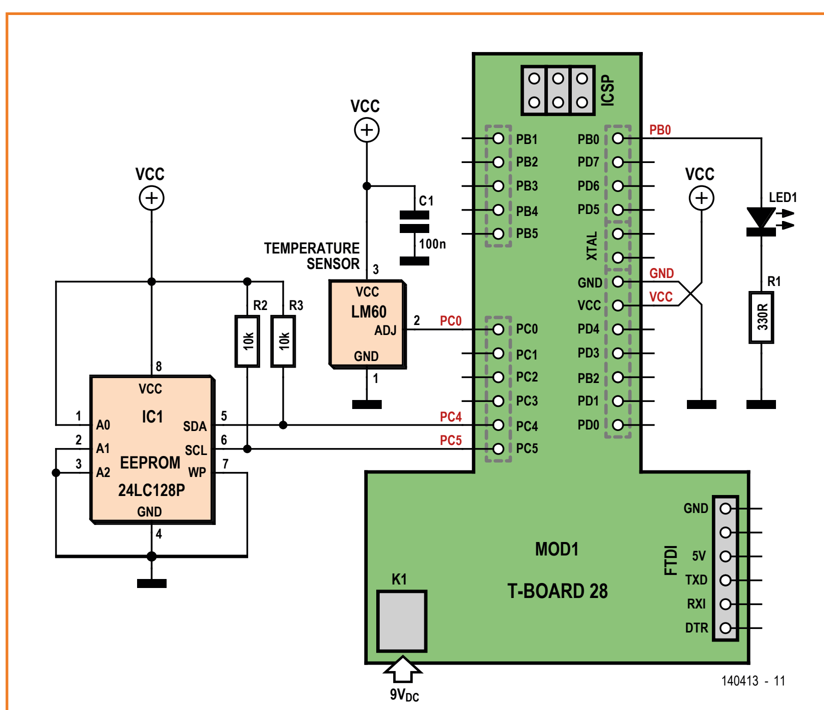


Figure 1.
“Circuigram” of our exercise aimed at getting T-Board 28 to draw as little current as possible while acting as a full blown Temperature Logger.

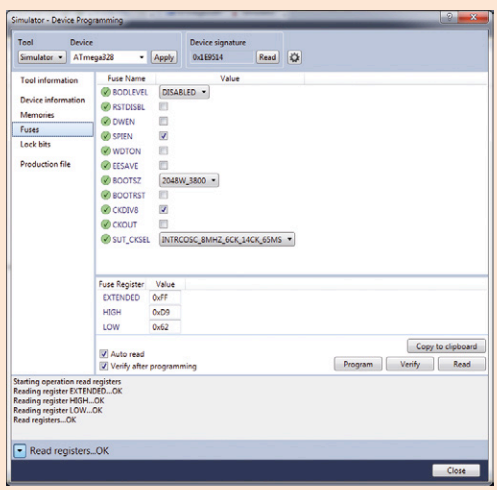
Don't Blow Your Fuse

Fuse words are used by AVR microcontrollers to configure a range of functions / options. They need to be flashed by an external ISP programmer, and once set they generally cannot be modified by the program running on the AVR (with a few exceptions).
They are often approached with great nervousness—and sometimes rightly so as setting invalid combinations can result in a "bricked" (no longer functioning) MCU. To "un-brick" the MCU, a special programmer is needed that operates at higher voltages.

There are three fuses on the T-Board 28's ATmega328—high fuse word (hfuse), low fuse word (lfuse) and extended fuse word (efuse). For our experiments, we will primarily alter the fuses that determine the clock source for the MCU—the lfuse. AVR microcontrollers can run off a range of internal and external clock sources, and we need to use the fuses to configure the MCU to use the ones that we want.

In order to calculate the values of the fuse words, you can wade through some fairly cryptic datasheet text—or use a fuse calculator. There are a number of online ones (I like the Engbedded one [4]) as well as apps you can install on your tablet/ smartphone.

Setting fuses using a programmer that is supported by Atmel Studio is very straightforward – you simply enter the fuse values in the Device Programming Dialog like so:



Setting Fuse Words in Atmel Studio.

If your programmer is not supported natively by Atmel Studio, then you will need to use AVRDude [5] to set the fuses:

1. Open a command prompt and navigate to the folder that AVRDude.exe is stored in

2. Enter the following to read the Fuse settings:
   `avrdude -c <programmer> -p <MCU> -U lfuse:r:-:h`

3. Enter the following to program a Fuse:
   `avrdude -c <programmer> -p <MCU> -v -U lfuse:w:0xFF:m`

where:

`<programmer>` is AVRDude's code for the specific programmer you're using (e.g. usbtiny for the USBTinyISP)—refer to the file avrdude.conf for a list of these;
`<MCU>` is AVRDude's code for the specific MCU being programmed (e.g. m328 for the ATmega328);
`lfuse` is the fuse being programmed—use lfuse, hfuse or efuse as appropriate;
`0xFF` is the value of the fuse being set—change this as necessary.

Finally, if you want to experiment more extensively with fuses, have a search online, there are a number of good resources.

Timing the Timer

Timer2 is an 8-bit timer, so can only count up to a value of 255. Each clock cycle adds to the timer until it reaches 255 where it can be configured to generate an interrupt. As you can see, if the Timer is running off a fast clock it won't take long to reach 255 and generate an interrupt: a 1-MHz clock will reach 255 after only 256 microseconds! Fortunately there are a couple of ways to slow the timer down. The first is to use a prescaler that divides the clock speed down further. The maximum prescaler available is 1024, which will result in a count of 255 being reached after 0.26 seconds. An improvement, but still too quick.

The second is to slow the clock speed down. We can't slow the main clock down, as we then aren't able to communicate over the serial port reliably. Timer2 however can work off an external crystal oscillator as well—in asynchronous mode (asynchronous because the external oscillator is not synchronized to the main CPU clock). The useful thing here is that very slow crystals can be used—I used a watch crystal at 32.768 kHz. With this slower crystal, the timer reaches 255 in 7.97 seconds, much more useful! Additionally, a slower crystal of course uses less power.

to print the log in comma-delimited format, or to clear the log. The program waits for 10 seconds to allow the user to make a selection, before entering the normal logging loop.
There are two versions of the code: Version 1 (**Figure 2**) shows the completely unoptimized code, while Version 2 (**Figure 3**) contains the far more power-efficient version. The optimization in the final version has been reasonably limited in order to ensure the code remains readable and easy to follow; there are certainly small additional power-savings that can be realized, and

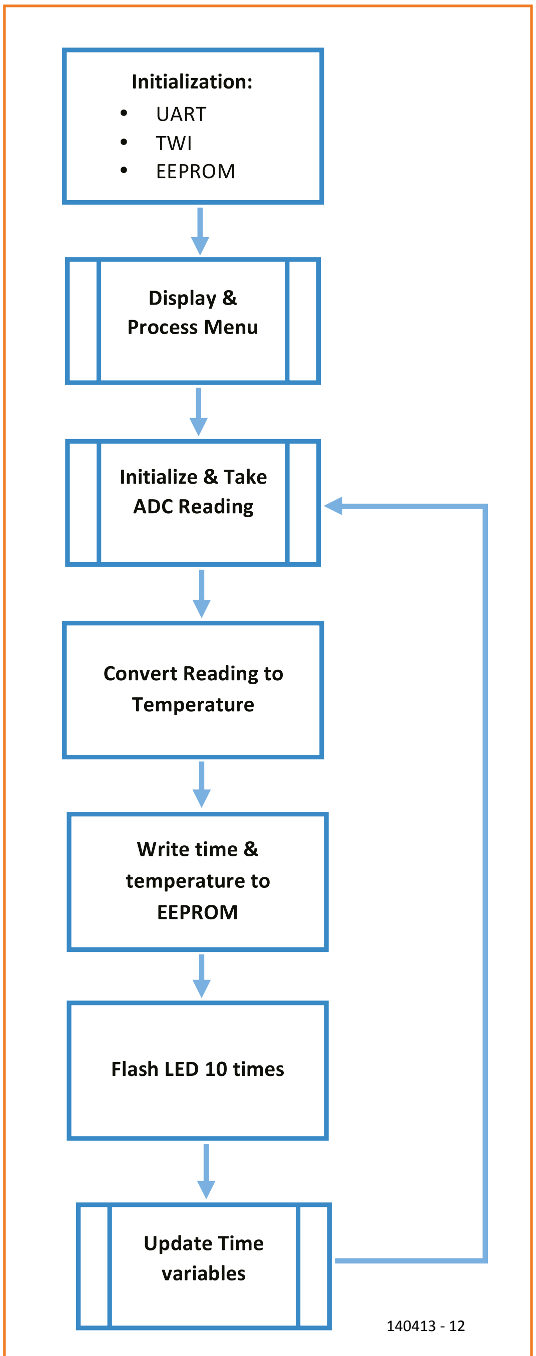


Figure 2. Flowchart of “version 1”, the entirely palatable but unoptimized firmware

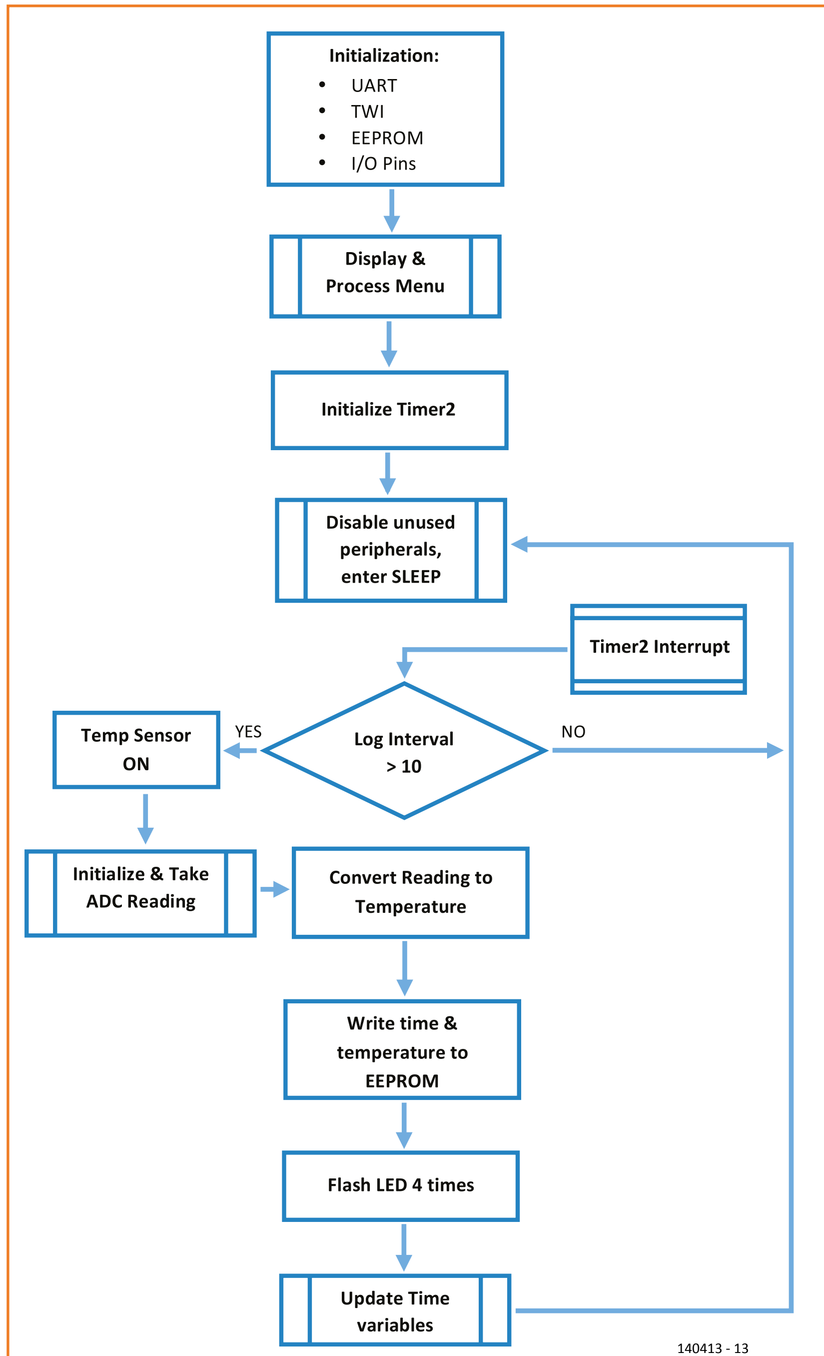


Figure 3. Flowchart of “version 2”, the T-Board 28 firmware optimized for current draw.

The Optimization Experiments

1. Establish a Baseline

Once the project is connected on a breadboard as set out in in Figure 1, the T-Board is initially configured in a similar way to an Arduino Uno. The input voltage should be set at 5 V using the voltage-selection jumper, and a 16 MHz external crystal oscillator connected. As we need to measure the current we'll remove the jumper completely and connect the DMM to the voltage-selection pins (black to the center pin, and red to the 5-V pin). In this configuration the DMM needs to be turned to the Amps setting in order to complete the circuit.

Next, the fuses need to be flashed to the values in **Table 1**. Finally, the "version-1" code (software download at [7]) needs to be loaded into Atmel Studio, compiled and then flashed to the microcontroller. This version uses a simple delay loop (implemented in the `_delay_ms()` function) to time the delays between readings. The flowchart in **Figure 2** gives an overview of the code.

The initial set of measurements show a consumption of approximately 12.4 mA while idle. From the code it can be seen that the LED flashes 10 times before entering idle state— measurements are taken once the DMM reading has stabilized after these 10 flashes.

A further single flash of the LED indicates that the temperature reading is about to be taken and written to the EEPROM—however the time taken to read the temperature turned out to be too short to register a meaningful reading on the DMM.

2. Reduce the Input Voltage

The simplest optimization yielded the greatest savings. Simply switch the voltage-selection jumper from the 5 V position to the 3V3 setting, and the T-Board now operates off 3.3 V. In our experiments of course we won't use the jumper, so switch the red lead of the DMM from the 5 V pin to the 3V3 pin.

The code needs to be changed so that the temperature calculations work off a reference voltage of 3.3 V; we need to change the compiler symbol `SYSTEM_MILLIVOLTAGE`. To do this, right-click on the project in the project browser and select properties. Then, on the *Toolchain* tab, click on *Symbols* under the *AVR/GNU C Compiler* section. Change the `SYSTEM_MILLIVOLTAGE` from 5000 UL to 3300 UL (the UL ensures the value is cast as an unsigned long). We then compile and flash the new code to the MCU.

The measured current at 3.3 V reads 5.76 mA, a reduction of 6.6 mA: that simple change has more than halved the power consumption of the logger!

3, 4, 5. Slow that CPU

These next three experiments examine the impact of slowing the CPU down, using a combination of swapping the external crystal oscillator and dividing the clock speed. With each change made, the compiler symbol `F_CPU` needs to be updated to reflect the effective clock speed in Hz as listed in **Table 2** (Note the spaces in the `F_CPU` column are there for ease of reading here, and should be removed when entered in Atmel Studio). For each of the rows in Table 2, follow the steps highlighted earlier: connect the programmer, update the fuses, change the `F_CPU` in the code, compile and flash the code to the MCU, disconnect the programmer and then switch the crystal if needed. As can be seen there is a clear correlation between the CPU speed and the current usage. Looking at the current usage at 500 kHz it looks like this is the path to choose. Unfortunately, the ATmega's USART is not able to reliably function at CPU speeds as low as 500 kHz (refer to the datasheet as well as [6] for useful BAUD rate accuracy calculator). So we need to go back a step.

6. Use the Internal Oscillator

Next I looked at the impact of using the internal oscillator at 8 MHz, and dividing by 8 to get an effective CPU speed of 1 MHz. This speed is sufficient to achieve reliable serial communications at low BAUD rates up to 4800—good enough for the simple menu at startup. Once the LFuse bits have been set to 0x62, the `F_CPU` symbol amended to 1000000UL, and the compiled program flashed to the MCU, then the external crystal can be removed.

Current draw was now slightly higher than at the clock speed of 500 kHz, but at 748 µA a great improvement on the original 12.36 mA!

7. Disable BOD

The ATmega328 has a Brown-out Detector (BOD), which allows a developer to guarantee consistent operation of a project by putting the MCU into a "reset" state if the input voltage drops below a specified level. However, this feature comes at a power cost. By setting the EFuse we can disable the BOD and squeeze those extra microamps out of the project. Set the EFuse to 0x07 to disable the BOD, and you'll note a small saving of 6 microamps—not a large saving, but every bit counts.

8. Sleep

It's now time to take a look at optimizing the code. In the version 1 (Figure 2), we used a delay loop to time the temperature readings. While it may seem like a delay loop is not doing anything, the CPU is in fact spinning its cogs counting the delay out. This of course uses power. Most microcontrollers allow you to put them to sleep (effectively turning the CPU and certain peripherals off) in order to save power, and then wake them up again when needed. Interrupts are used to wake the CPU up—either based on changes on particular pins, incoming communications, or specific elapsed time. The AVR is no exception, and has 6 different sleep modes with differing levels of power saving. I won't go into these in detail here, other than to say that I'll use the mode called (imaginatively!) "Power-save". This mode allows us to use Timer2 to wake the CPU up after a specific interval—Timer2 being the only timer that can in fact wake the CPU. In order to save on power and work with more manageable

timings, Timer2 is set to run off an external watch crystal. Refer to the **text box** "Timing the Timer".

To put this experiment in action, no Fuse bits need to be set. A 32.768 kHz watch crystal needs to be inserted carefully in the T-Board 28—these leads are very thin so make sure that you don't damage them, while at the same time ensuring a good contact with the header on the T-Board.

Then load, compile and flash the Version 2 code onto the MCU. Version 2 does less flashing of the LED, just four flashes at the end of each temperature reading. I took my readings once the current stabilized after the four flashes, and was delighted to see a measurement of only 59 µA. This kind of current draw should allow an 800-mAh battery to power the project for nearly a year and a half. But there is still more we can do!

9. Shut Down Unused Peripherals

The ATmega328 has a register that allows you to power off unused peripherals – the Power Reduction Register (PRR). The function `reducePower()` disables the Timer0, Timer1, SPI and the USART. The TWI and Timer2 are left enabled as we need them to communicate with the EEPROM and wake the MCU from sleep respectively. The ADC is already managed as part of the conversion process, so no further optimization needed there. Additionally this function enables the pull-up resistors on the unused pins to prevent them from floating—floating pins can result in additional current draw. With the DMM I was using I wasn't able to see any additional power saving from using these settings—looking at the datasheet percentages it seems that the real saving is achieved when the CPU is in active mode as the percentage savings are quite low.

10. A Final Change

After some comparison to the datasheet I wasn't happy with the power consumption I was achieving. Of course the MCU isn't the only element drawing current—the EEPROM and the LM60 also have an impact. A few quick tests, and references to their datasheets, and I noted that the EEPROM only draws 100 nA in standby mode while the LM60 has a typical quiescent current draw of around 82 µA. By altering the circuit so that the LM60 was powered off the T-Board's PD7 (well within the pin's maximum output current of 20 mA), and modifying the code to power the sensor only for the period needed to take a reading, I managed to get the current draw down to an incredible 1 µA (the lowest my DMM could measure). After seeing this reading I popped the champagne and considered the experimental process a success! I took a few additional measurements, and I arrived at an estimated peak current draw of approximately 65 µA during temperature measurement. I rounded this up to 1 mA for good measure, and then by calculating a few averages I arrived at an estimated battery lifespan of six years on an 800 mAh battery. Six years on a single battery is more than enough for our remote sensor.

Table 1. Results of Experiments

Experiment	Description	F_CPU	SYSTEM_ MILLIVOLTAGE	LFUSE	EFUSE	Current (mA)
1	16 MHz and 5 V	16 000 000 UL	5000 UL	0xFF	0x05	12.360
2	Reduce to 3.3 V	16 000 000 UL	3300 UL	0xFF	0x05	5.760
3	Use 8 MHz External Crystal	8 000 000 UL	3300 UL	0xFF	0x05	3.360
4	Use 4 MHz External Crystal	4 000 000 UL	3300 UL	0xFD	0x05	2.130
5	Divide Clock by 8	500 000 UL	3300 UL	0x7D	0x05	0.560
6	Internal 8 MHz oscillator + Divide by 8	1 000 000 UL	3300 UL	0x62	0x05	0.748
7	Disable BOD	1 000 000 UL	3300 UL	0x62	0x07	0.742
8	Sleep Mode with Timer2	1 000 000 UL	3300 UL	0x62	0x07	0.059
9	Shut down unused peripherals	1 000 000 UL	3300 UL	0x62	0x07	0.059
10	Disable temp sensor when not used	1 000 000 UL	3300 UL	0x62	0x07	0.001

Table 2. Xtal parameters for Experiments 3, 4, 5.

Experiment	Crystal	Divide by 8?	Effective clock speed	F_CPU	LFUSE	Current
3	8 MHz	No	8 MHz	8 000 000 UL	0xFF	3.36 mA
4	4 MHz	No	4 MHz	4 000 000 UL	0xFD	2.13 mA
5	4 MHz	Yes	500 kHz	500 000 UL	0x7D	0.56 mA

Web Links

[1] T-Boards 8/14/28, Elektor September 2014, www.elektor-magazine.com/130581

[2] Atmel Studio: www.atmel.com/atmelstudio

[3] USB Tiny: www.crash-bang.com/using-usbtiny-with-atmelstudio/

[4] Fuse Calculator: www.engbedded.com/fusecalc/

[5] AVRDude: http://savannah.nongnu.org/projects/avrdude

[6] Baud Calculator: www.wormfood.net/avrbaudcalc.php

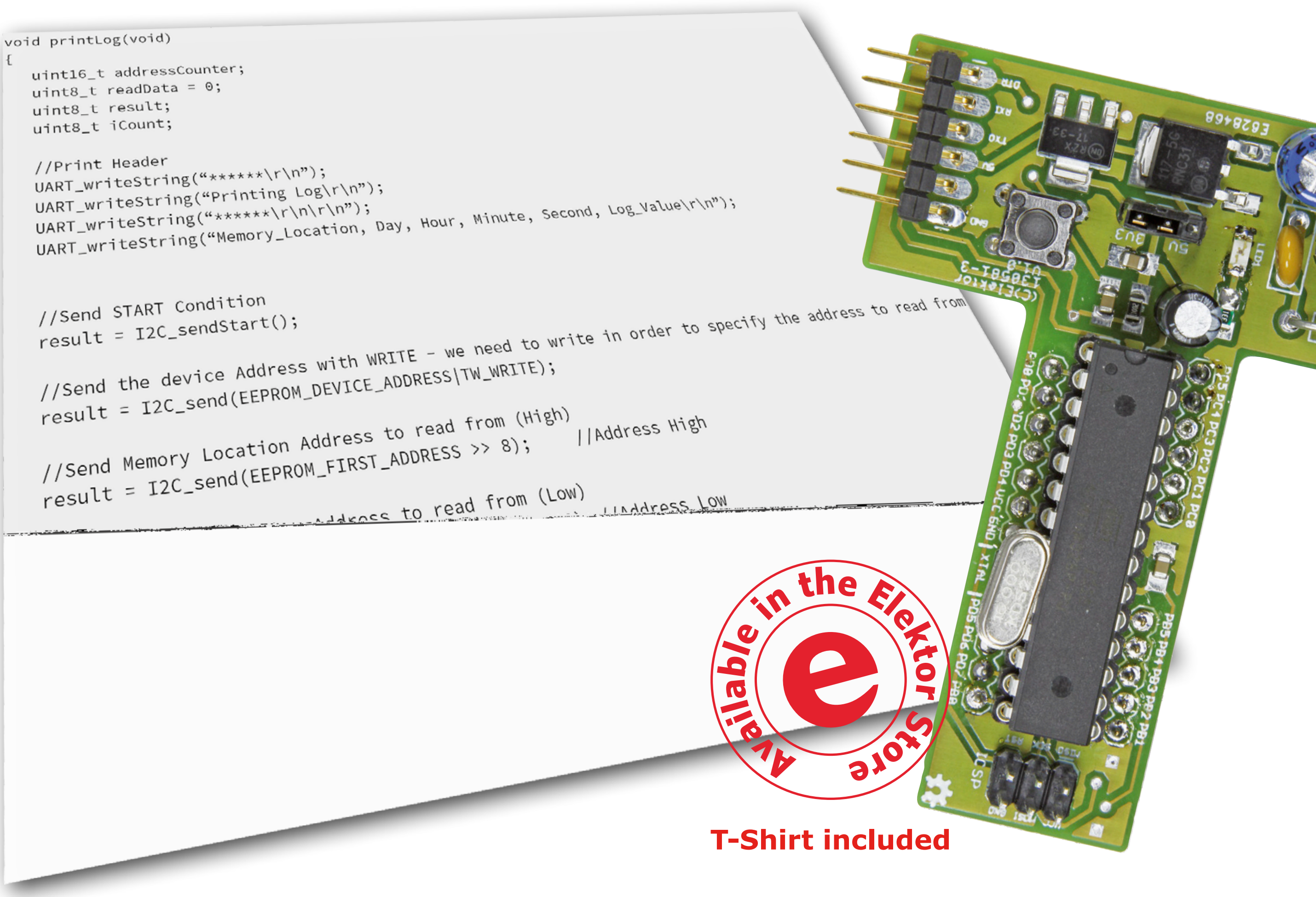
[7] Version 1 & 2 download: www.elektor-magazine.com/140413

I challenge readers to take it further and share their thoughts on www.elektor-labs.com and http://forum.elektor.com (Topic: Microcontrollers & Embedded).

T-Board helps with optimization

In order to reduce current consumption on a microcontroller project, there are (at a basic level) four common areas to focus on. The first is to reduce the input voltage, resulting (simplistically) in a physical illustration of Ohm's Law. The second is to reduce the clock speed of the CPU, as less clock cycles per second result in lower current draw. The third is to put the CPU into a sleep mode when it isn't needed. Finally, disabling unused peripherals and components will result in further reductions in power.

The T-Board 28 has been designed with the flexibility to help with the first and second items discussed above: the ability to select input voltage, and a means to switch out crystal oscillators to change the CPU clock speed. The second and third items are implemented in the pro-



gram code. Finally, the voltage selector of the T-Board acts as useful point to measure the current usage in order to assess the effectiveness of the optimizations.

In addition to the items outlined above, a number of other efficiencies can be gained—for example more efficient coding, which results in fewer clock cycles to achieve specific tasks, will reduce the current requirements. As discussed earlier, the code efficiency has not been a focus of this project—optimizing your coding in C/C++ is a whole series of articles on its own!

Measuring the current consumption

It is assumed that many readers using the T-Board (including myself) may not have access to a fully kitted lab. Therefore I used a relatively crude, but nevertheless effective, means of measuring the current usage—a low-cost digital multimeter (DMM). The DMM was switched to measure amps, and the probes then connected (using female connectors) to the voltage-selection header pins on the T-Board. This allowed me to measure the current usage of the T-Board at a point in time. The positions of the jumpers in the circuit isolate the draw of the project, and are not impacted by the power supply circuitry.

In order to measure the current more accurately, I desoldered and removed the power-indicator LED as I found its current draw far outweighed that of the MCU itself. This also reduced the consumption sufficiently at higher levels of optimization to enable me to switch my DMM to the µA setting, allowing me to better measure the impact of some of the smaller changes.

Of course the limitation with this measurement method (apart from a reduced level of accuracy) is that the current is only measured at a point in time. Ideally one would like to record the measurements over time in order to calculate an average current draw (this is in itself a useful project to include in a future edition of Elektor). This is where the LED connected to PB0 came in useful as an indicator of the activity of the program, and therefore a context to the current being measured.

The experiments

The spread on the previous pages of this article focus on the experiments carried out to reduce current draw, detailing what changes were made and the measured impact of these. In order to implement each of the progressive optimizations, these steps were followed:

1. the ISP programmer was connected and any changes to fuses made;
2. any changes to the code were compiled and flashed to the T-Board;
3. the ISP programmer was disconnected;
4. any physical changes to the circuit were made;
5. a 9-V battery was connected to the 2.1-mm jack and the current measured with the DMM.

Any FTDI cables/boards should not be connected when the measurements are taken.

The results of the various experiments are summarized in **Table 1**, along with the changes made. Before we get started, a quick note on the programmer: Atmel Studio [2] supports a number of ISP programmers natively, which require little or no special configuration. There are also a number of other programmers (usually available at a lower cost) that are not natively supported but can be configured to work under Atmel Studio (eg. USBTiny, USBasp, etc.). Apart from the flashing of the program onto the MCU, a key difference is the way the fuses need to be set (Refer to the **text box** “Don’t Blow Your Fuse” for more information). There are various resources online explaining how to configure Atmel Studio to support these third-party programmers [3].

Next Steps

I found this process enormously helpful, and it really highlighted to me the benefit of a flexible, lean & mean platform like the T-Board. Overall, consumption was reduced from an Arduino-class 12.3 mA to an average of 13 µA (averaged based on time spent in sleep vs. active mode); a reduction of 99.89%!

I challenge other readers to take this to the next level—perhaps migrating to the T-Board 8, or including an RF sensor to transmit the readings. Either way, with your t-shirt on.

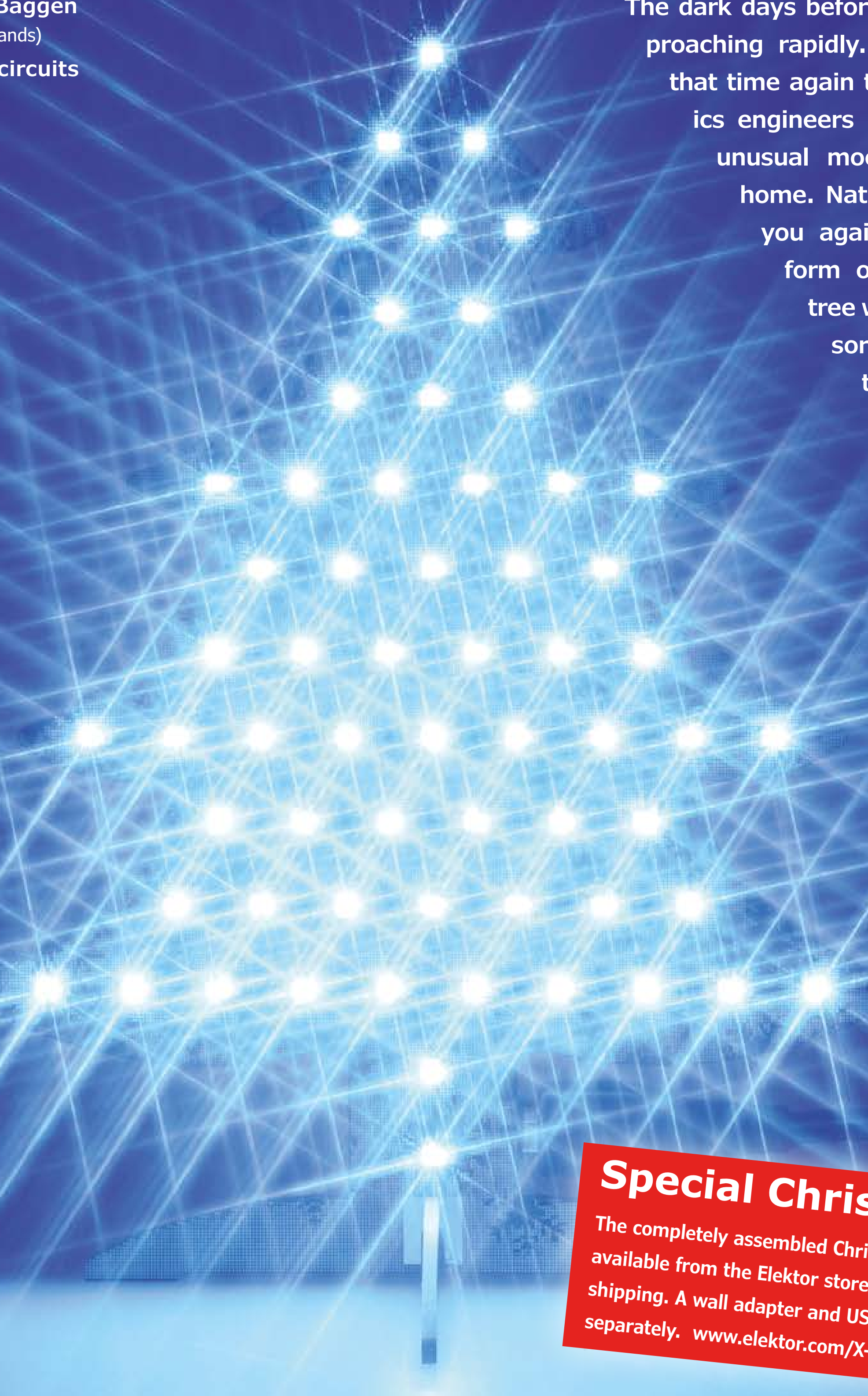
(140413)

Programmable Christmas Tree

Brings (blue) light into the darkness

Text: **Harry Baggen**
(Elektor Netjerlands)
Design: **Eurocircuits**

The dark days before Christmas are approaching rapidly. So it has become that time again that we as electronics engineers provide some more unusual mood lighting for the home. Naturally we will assist you again this year, in the form of a nice Christmas tree which conjures up all sorts of beautiful patterns on its 62 LEDs.



Electronics hobbyists are always keen to make a festive gadget for the celebrations around this time of the year. Not something read-built from a garden center or home improvement store, but a circuit which has something special and that you can't just buy anywhere. Over the years Elektor have published numerous Christmas circuits, the majority of them projects that contain a number of LEDs which provide the ambiance.
This year, in conjunction with the printed circuit board manufacturer Eurocircuits, we have designed a very nice circuit for you, which you can build yourself of course if you have the time and inclination, but we also offer this Christmas tree ready-built in the Elektor web store for an attractive price. You can always tell your visitors that you soldered the teeny-weeny SMD components in the little tree yourself with the aid of a microscope and a very small soldering iron during the annual Christmas function at your work...

The little Christmas tree that you see here is provided with no less than 62 bright blue LEDs on the front that, with their color alone, will already generate a pleasant Christmas atmosphere. The control electronics is on the back, mainly comprising a powerful microcontroller. True, we have had other flashing Christmas trees in our magazine before, but this particular specimen has an extensive light show built in, which causes all kinds of patterns or texts to 'run' across the LEDs. A number of patterns have already been pre-programmed, but a great feature is that you can very easily design your own patterns and effects using your PC and then simply store them in the memory of the Christmas tree via the USB connection.

Powerful microcontroller

An ARM Cortex-M0 microcontroller from STM has been used for controlling the LEDs (see the schematic in Figure 1). This relatively cheap 32-bit

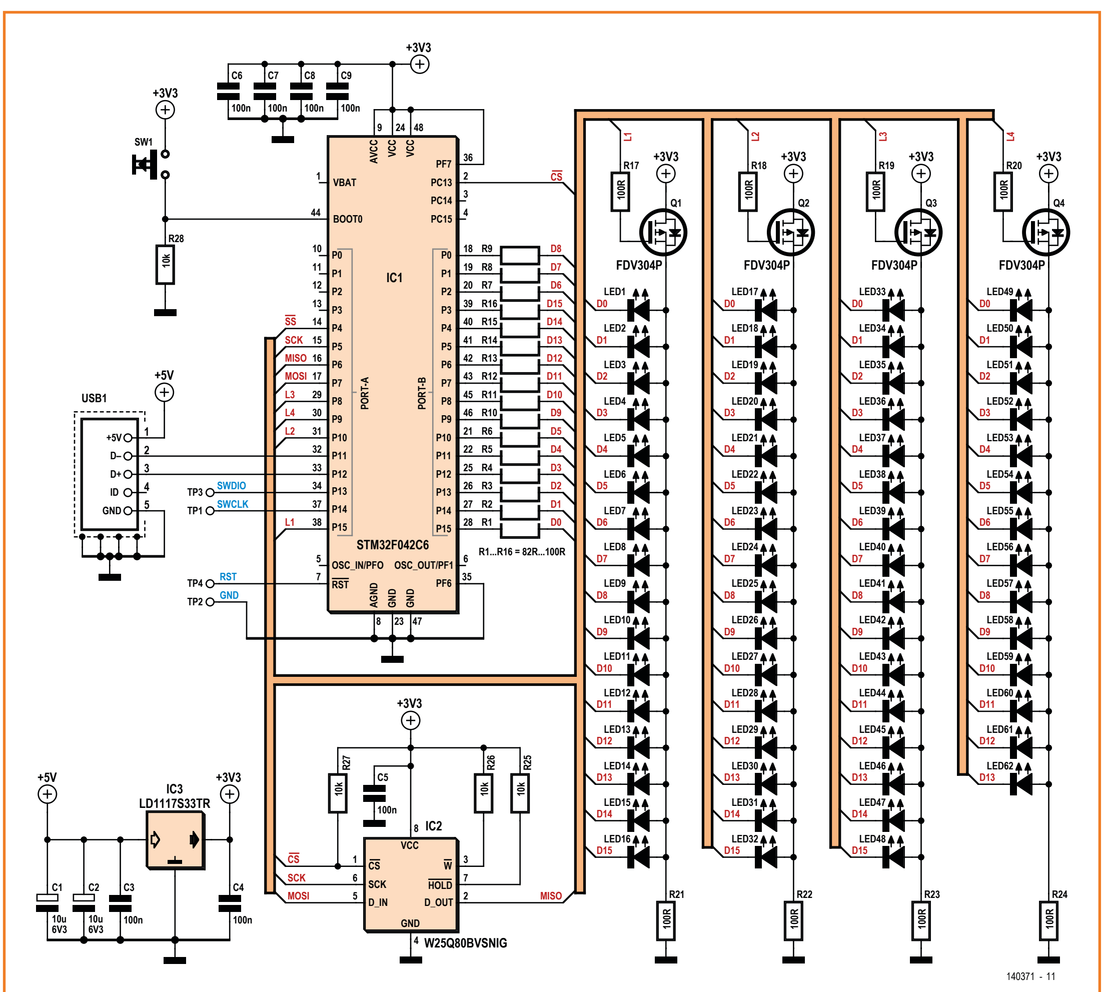


Figure 1.
The schematic for the Christmas tree mainly consists of a microcontroller and 62 blue LEDs.

Component List

Resistors
R1–R16 = 82–100Ω (SMD 0805)
R17–R24 = 100Ω (SMD 0603)
R25–R28 = 10kΩ (SMD 0603)

Capacitors
C1,C2 = 10µF 6.3V tantalum (SMD A)
C3–C9 = 100nF 10V ceramic (SMD 0603)

Semiconductors
LED1–LED62 = LED blue (SMD 1206)
Q1–Q4 = FDV304P (SOT23)
IC1 = STM32F042C6 (LQFP48)
IC2 = W25Q80BVSNIG (SO8)
IC3 = LD1117S33TR (SOT223)

Miscellaneous
USB1 = micro USB connector, PCB mount (47346-0001)
SW1 = pushbutton, PCB mount (TACTB-64K-F)
PCB (files only) # 140371-11

microcontroller offers, besides much computing power, an integrated USB2.0 interface. This allows for a very easy connection of the microcontroller to a PC. Thanks to the built-in bootloader it is very straightforward to download new LED-patterns or new firmware. In order to provide the user which a generous amount of space for patterns or animations of their own design, a serial flash memory with a capacity of 1 MB (8x1 Mbit) is connected to the microcontroller. This may appear small compared to the typical size of memory sticks and memory cards these days, but nevertheless provides sufficient space to store more than 90,000 complete LED patterns.

The LEDs are driven in a 4x16 matrix (not entirely, 3x16 + 1x14). Therefore 16 (14) LEDs are driven alternately by port pins P0 through P15 of the controller. The four columns are continually multiplexed. Because of the persistence of vision of the human eye, this fast switching is not noticed and it appears that all the LEDs are on continuously. The LED columns are switched by four P-channel MOSFETs of the type FDV304P. The cathodes of the LEDs are connected via current-limiting resistors (R1 through R16) directly to the port output pins of the microcontroller. This saves a large number of components, but we do have to take into account that the maximum

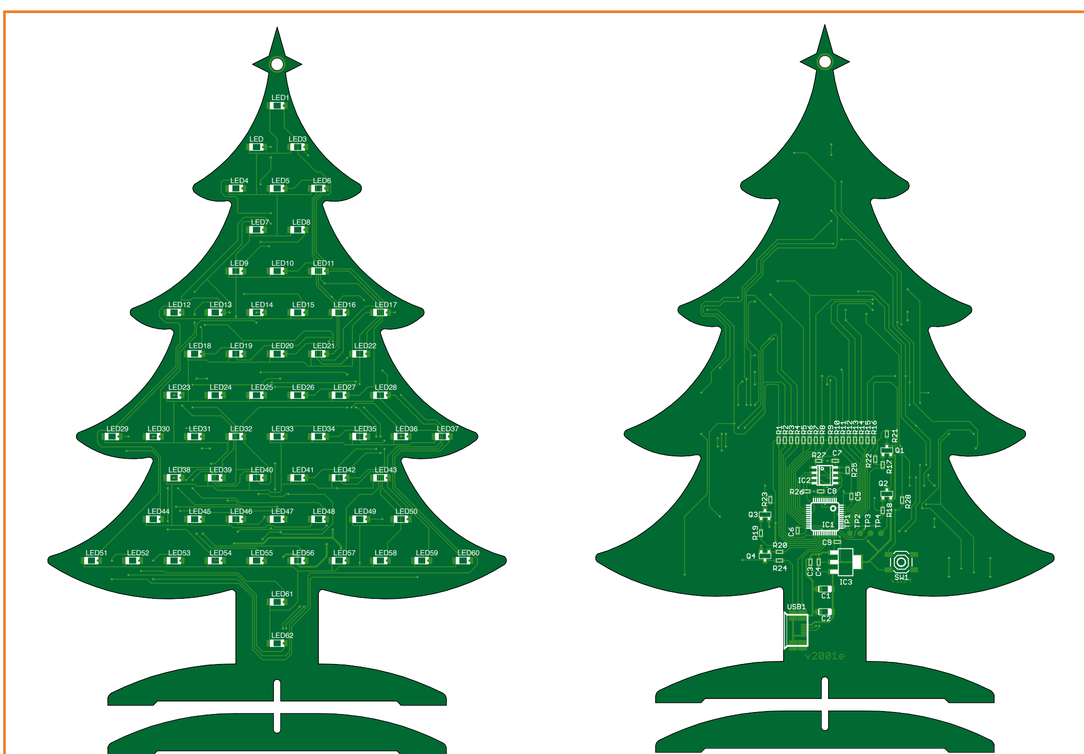


Figure 2.
The circuit board is double-sided and screened on the front with a so-called PCB-Pixture solder mask [1] depicting snowflakes.
(60% of real size)

total current that the microcontroller can handle is only 80 mA, which therefore means a maximum current of 5 mA per LED. If we use LEDs with sufficient efficiency then they will still be very visible in a reasonably bright environment. The entire circuit is powered from the micro-USB connection. A 3.3-V regulator (IC3) provides a regulated power supply voltage for the microcontroller and the LED matrix.

Figure 2 shows the printed circuit board design for the little tree. The LEDs are on the front, the remaining components including the micro-USB connector are on the back. For this occasion the circuit board is screened with a PCB-Pixture solder mask [1] depicting snowflakes.

For those of you who are keen to etch the circuit board yourself and populate it with components we feel obliged to warn you that some of these SMD components are difficult to solder by hand!

The source- and hex code files are available from the Elektor website [2]. The ARM software is written in C using the Keil compiler. The largest part of the code is related to the USB communications and this is based on the ST firmware library.

The bootloader mode will be activated when the pushbutton on the back is pressed while the tree is connected to a PC. It is then possible to download or update the firmware with the aid of the ST software DfuSe [3].

Software

By default, the ready-built Christmas tree from the Elektor shop is already provided with a wide range of animations. If you are happy with those then there is nothing more you need to do. If, however, you would like to create your own patterns, animations or running texts and whatever else you can come up with then we direct you to a special interactive web page that we have built for you where you can indulge yourself [4]. This page contains a picture of the Christmas tree with all its LEDs. You can make your own animations by using your mouse to click the LEDs on the screen on or off. This way you create a pattern, where you can also select the desired duration and brightness. All generated patterns appear at the bottom of the screen next to each other. There is also the possibility to enter a scrolling text (text symbol at top left). In this way you can build your own animations. You can store an animation for later use. Using the preview button you can view the animation on the screen.

Program your own animations and win!

Eurocircuits have organized a programming competition around this little Christmas tree. Put your own animation on the web page for the Christmas tree [4] and make sure that as many people is possible 'like' your animation. For those with the most 'likes' there are several nice prizes to be won! The closing date for this competition is January 7, 2015—winners will be notified personally.

You can make multiple of these animations and 'thread' these together into one long light show. Once that is done you can download the entire sequence to the Christmas tree.

The tree is connected to a PC using a micro-USB cable. When you do that for the first time Windows will install a driver. Windows recognizes the little tree as a HID peripheral. After that you can 'flash' the new animation to the tree from the web page.

For use in the living room you can connect the tree to a 5-V wall adapter which is fitted with a micro USB cable.

Enjoy and many happy returns.

(140371-I)

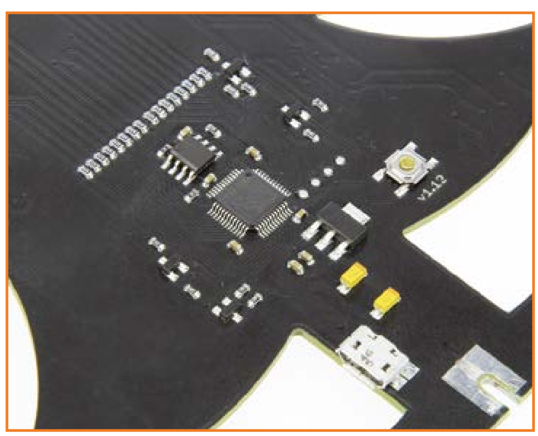


Figure 3.
The components mounted on the back of the Christmas tree.

Web Links

[1] www.eurocircuits.com/blog/171-PCB-PIXture-launched

[2] www.elektor-magazine.com/140371

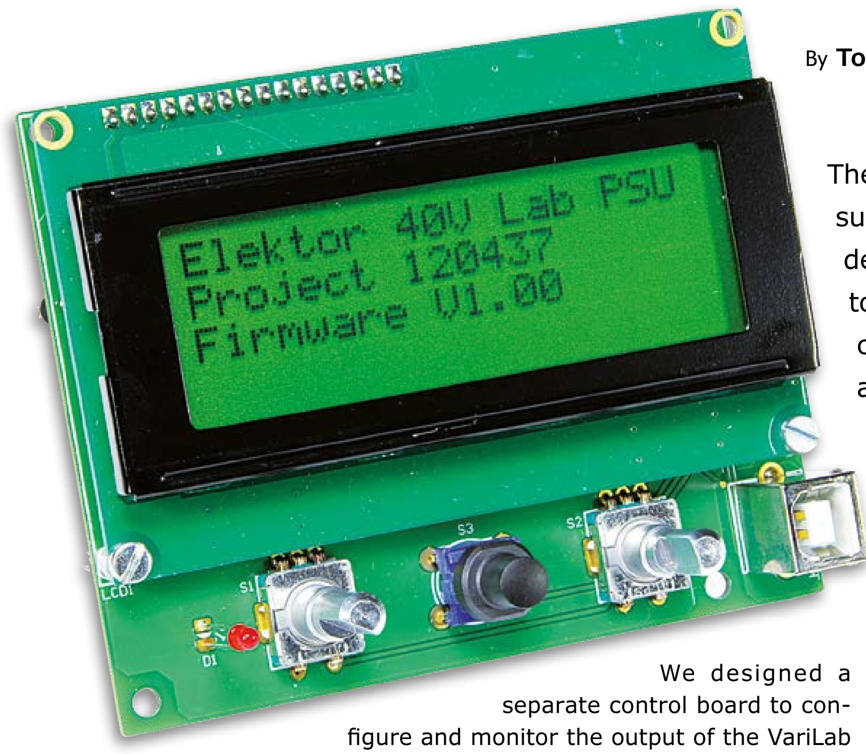
[3] www.st.com/web/en/catalog/tools/FM147/CL1794/SC961/SS1533/PF257916

[4] www.eurocircuits.com/x-mas

VariLab 402 (2)

Control and display board

By **Ton Giesberts** (Elektor Labs)



The control portion of the VariLab power supply uses a microcontroller to set the desired output voltage and current and to measure all significant voltages and currents in the circuit. All measurements and settings are shown on a four-line LCD module. Naturally, there is also a USB port for communication with a PC.

We designed a separate control board to configure and monitor the output of the VariLab power supply board described in the previous issue and display the output voltage and current as well as other data. Its main components are an Atmel ATxmega128A4U-AU microcontroller and an LCD module with four lines of twenty characters. The microcontroller has onboard ADCs and DACs for setting and measuring the output voltage and current, so the component count can be kept relatively small.

Circuit description

As you can see from the schematic diagram (**Figure 1**), the circuit is built around an ATxmega microcontroller (IC1). The balanced supply voltages for the entire circuit (±5 V) are taken from the supply board. The microcontroller operates from a 3.3 V supply voltage, which is provided by voltage regulator IC3.

All measurement and control signals are present on connector K3. The circuit also receives the necessary supply voltages through this connector. The sense signals for the output voltage and current of the supply board are fed to the ADC3 and ADC4 inputs of the ATxmega, with the current sense signal first buffered by IC2d. Resistor R5 is dimensioned so that the range of the sense signal for the power supply output voltage is virtually the same as the measuring range of ADC3. The voltages Vsmps' and 48Vin are also routed to the control board and measured by ADC5 and ADC6. Resistor R6 is dimensioned to apply 5% of Vsmps' to the ADC5 input so that the input signal level remains within the measuring range. Diodes D2–D5 protect the ADC inputs against overvoltages and negative voltages.

Two signals are necessary to set the output voltage and maximum output current of the power supply: 0–4 V for the voltage and 0–2 V for the current. The reference voltage for the ADCs is provided by an external 2.5 V reference source (IC6, for which we chose the old standby LM336) because the internal reference voltage of the microcontroller is not sufficiently accurate. The reference voltage can be adjusted precisely with P1, but this trimpot can usually be left at its maximum setting because the calibration can also be done in software. The output voltages of the DACs are amplified by a factor of 2 by a pair of opamps (IC2A and IC2B), which in theory yields

a maximum control voltage of 5 V.

A disadvantage of using the internal DACs is that the output range does not extend all the way to 0 V (the data sheet states a minimum value of 0.15 V). Since we definitely want to be able to adjust the output voltage down to 0 V, an offset voltage of 0.3 V is applied to the two opamps to ensure that the control voltage range on the opamp outputs can start at 0 V. The 0.3 V offset voltage is derived from the 2.5 V reference voltage by R11/R12 and buffered by IC2c.

For the opamps we chose a Microchip MCP6004-E/SL quad device, which features rail-to-rail inputs and outputs. This IC is powered from +5 V and -0.6 V (the latter via D8) because its maximum rated operating voltage is 6 V. This arrangement allows the output voltage to be adjusted all the way down to zero.

The four-line LCD module is driven in 4-bit mode by the ATxmega. The display is powered from 5 V. Although the logic high level of the drive signals from the microcontroller is only 3.3 V, that is adequate for reliable operation. The display contrast can be set with P2. The microcontroller can switch on the display backlight via T1 when a jumper is fitted on JP3. Since the voltage regulator on the supply board that provides the+5 V supply voltage does not have an extra heat sink, it is not advisable to leave the backlight on all the time. In the current version of the firmware the LCD is only illuminated when current limiting is active. The backlight can be permanently disabled by removing the jumper at JP3.

A pair of rotary encoders with built-in pushbuttons (S1 and S2) and a separate pushbutton (S3) are provided for the power supply operator interface. The separate pushbutton can be used for several functions, including quickly switching the power supply on or off. By default the output voltage is set with S1 and the output current is set with S2.

Figure 1.
Schematic diagram of the control and display board. All of the measurement and computation tasks are handled by an ATxmega128 microcontroller.

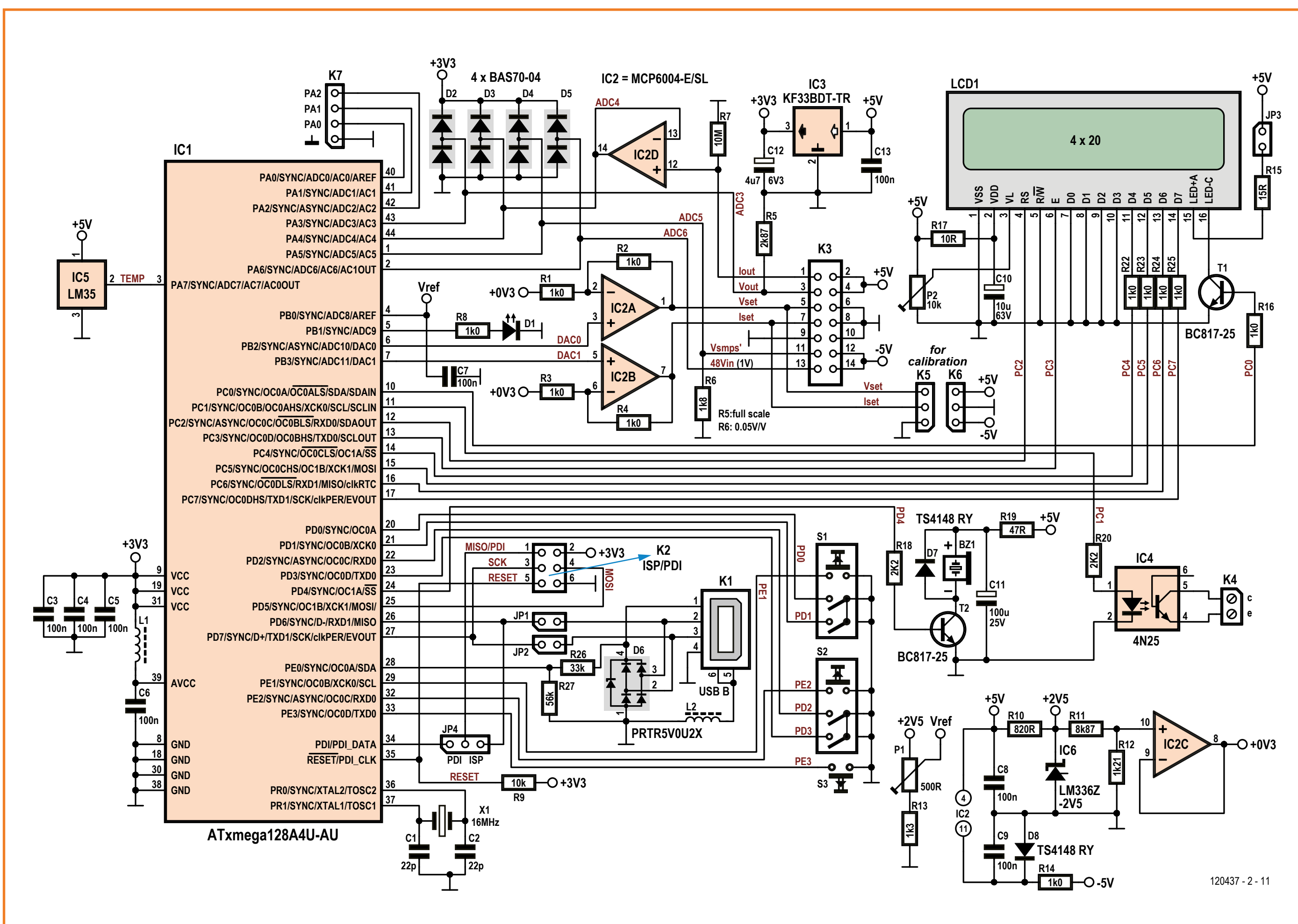
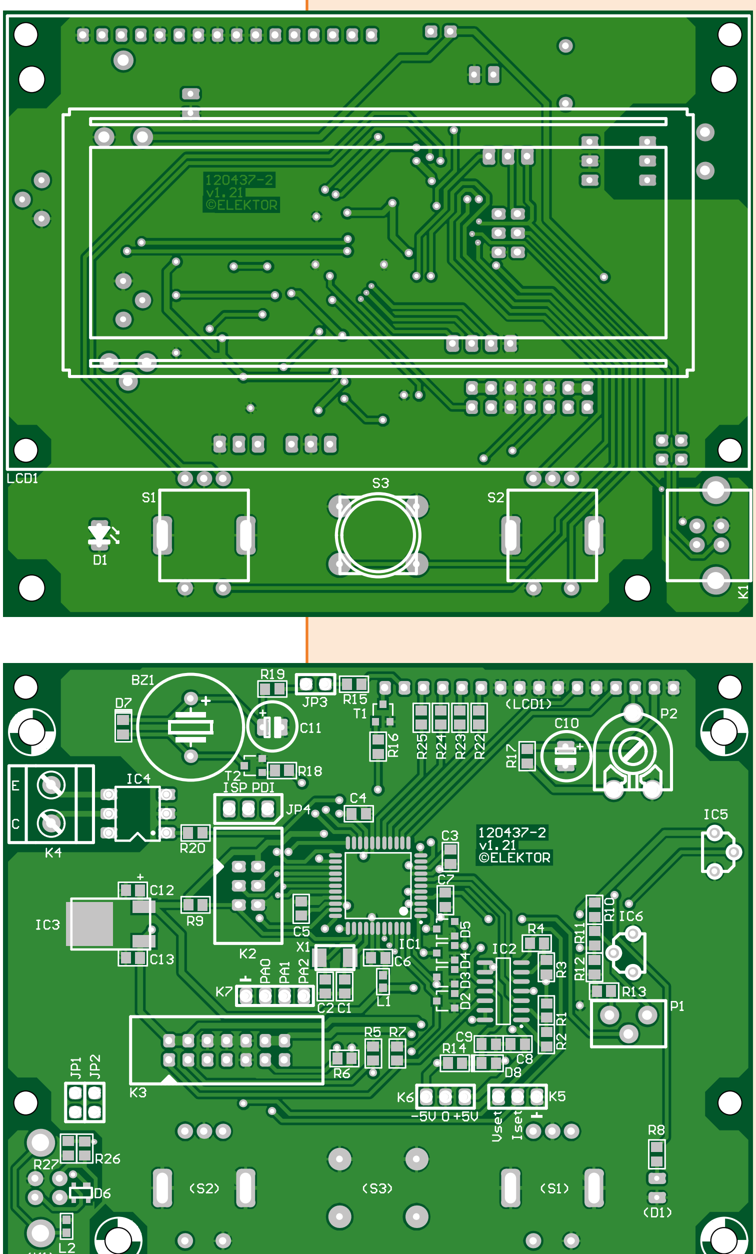


Figure 2.
The component layout of the double-sided control board.



Component List

Resistors

Default 0.125W, SMD 0805)
R1–R4,R8,R14,R16,R22–R25 = 1kΩ 1%
R5 = 2.87kΩ 1%, 0.1W
R6 = 1.80kΩ 1%
R7 = 10MΩ 1%
R9 = 10kΩ 5%
R10 = 820Ω 1%
R11 = 8.87kΩ 1%
R12 = 1.21kΩ 1%
R13 = 1.30kΩ 1%
R15 = 15Ω 1%, 0.25W
R17 = 10Ω 5%
R18,R20 = 2.2kΩ 5%
R19 = 47Ω 5%
R21 = not fitted
R26 = 33kΩ 5%
R27 = 56kΩ 5%
P1 = 500Ω 10%, 0.5W, 25-turn preset, top adjustment (Bourns 3296Y-1-501LF)
P2 = 10kΩ 20%, 0.15W, preset, top adjustment

Capacitors

C1,C2 = 22pF 50V, 5%, SMD 0805 C0G/NP0
C3..C9,C13 = 100nF 25V, 10%, SMD 0805 X7R
C10 = 10µF 63V, 20%, diam. 6.3mm, 0.1” pitch
C11 = 100µF 25V, 20%, diam. 6.3mm, 0.1” pitch
C12 = 4.7µF 6.3V, 10%, SMD 0805 tantalum (case R, AVX TAJR475K006RNJ)

Inductors

L1,L2 = 330Ω @ 100MHz, 0.08Ω 1.7A, SMD 0603

Semiconductors

D1 = LED, red, 3mm, wired
D2,D3,D4,D5 = BAS70-04 (SMD SOT-23)
D6 = PRTR5V0U2X (SMD SOT-143B)
D7,D8 = TS4148 RY (SMD 0805)
T1,T2 = BC817-25 (SMD SOT-23)
IC1 = ATxmega128A4U-AU (SMD TQFP-44), programmed, Elektor Store #120437-41)
IC2 = MCP6004 (SMD SOIC14)
IC3 = KF33BDT-TR (DPAK-3)
IC4 = 4N25 (DIP-6)
IC5 = LM35 (TO-92)
IC6 = LM336Z-2V5 (TO-92)

Miscellaneous

K1 = USB connector, type B, PCB mount, wired, vertical model (Lumberg 2411 01)
K2 = 6-pin (2x3) pinheader, 0.1” pitch
K3 = 14-pin (2x7) pinheader, 0.1” pitch
K4 = 2-way screw terminal block, 0.2” pitch
K5,K6,JP4 = 3-pin pinheader, 0.1” pitch
K7 = 4-pin pinheader, 0.1” pitch
JP1,JP2,JP3 = 2-pin pinheader, 0.1” pitch
jumpers for JP1,JP2,JP3,JP4
S1,S2 = 18PPR rotary encoder with integrated pushbutton (e.g. Alps EC11E183440C)
S3 = pushbutton, PCB mount (e.g. Multimec RA3FTL6)
S3 = cap for pushbutton, h=19mm (e.g. Multimec 1S09-19.0)
BZ1 = piezo buzzer, 1.5–16 V_{DC}, max. 8 mA, diam. 14mm, 7.62mm pitch
X1 = 16MHz quartz crystal, C_L 18pF, dim. 5 x 3.2mm (Abracon ABM3-16.000MHZ-B2-T)
LCD1 = 4x20 characters, with backlight, Elektor Store #120061-73
PCB # 120437-2

Buzzer BZ1, which is driven by the microcontroller through T1, generates an acoustic signal in situations such as current limiting. Indicator lamp LED D1 is lit when the power supply output is enabled. Temperature sensor IC5 measures the temperature in the enclosure, which can be displayed on the LCD. The sensor is located directly on the PCB, and when the board is installed as intended in an enclosure the sensor is close to the heat sink on the supply board. For other applications the sensor can also be mounted somewhere else and connected to the board by a few wires. There is also a galvanically isolated output (IC4) available for switching purposes, although this capability is not used in the current version of the firmware.
A USB port for connection to a PC is additionally present. This can be used to send measurement data to the computer, or the voltage and current can be set from the computer with the aid of a terminal emulator program. The USB port is normally connected directly to the microcontroller over JP1 and JP2, but if desired it can be fully isolated by removing these two jumpers. Diode D6 protects the microcontroller against potential overvoltages on the USB port. Voltage divider R26/R27 is included to prevent the microcontroller program from hanging if the USB connection is suddenly broken.
The microcontroller can be put into programming mode (ISP or PDI) by fitting a jumper on JP4. PDI is usually the better option, and it is also supported by the Atmel AVRISP mkII programmer. With both settings the microcontroller is programmed over connector K2.
The voltages Vset, Iset and ground are present on connector K5 for convenient measurement during calibration. If the control board is used in a project other than the VariLab power supply, the balanced ±5 V supply voltages for the control board can be connected to K6. The maximum current consumption is +50/−5 mA with the backlight off or +105/-5 mA with the backlight on. Finally, K7 provides access to a number of free I/O pins of the microcontroller (PA0, PA1 and PA2).

Construction

The PCB layout shown in **Figure 2** is compact and just big enough to hold the four-line display module, the rotary encoders and the pushbutton (see also **Figure 3**). Components are mounted on both sides of the board. Most of the components are on the component overlay side, including all of the SMDs, connectors and leaded components. The only components mounted on the other side are the pushbutton, the two rotary encoders, the vertical USB connector and the indicator LED, and of course the display module.
Be sure to use the components shown in the components list when assembling the board. Substitutions are not allowed. The comments in

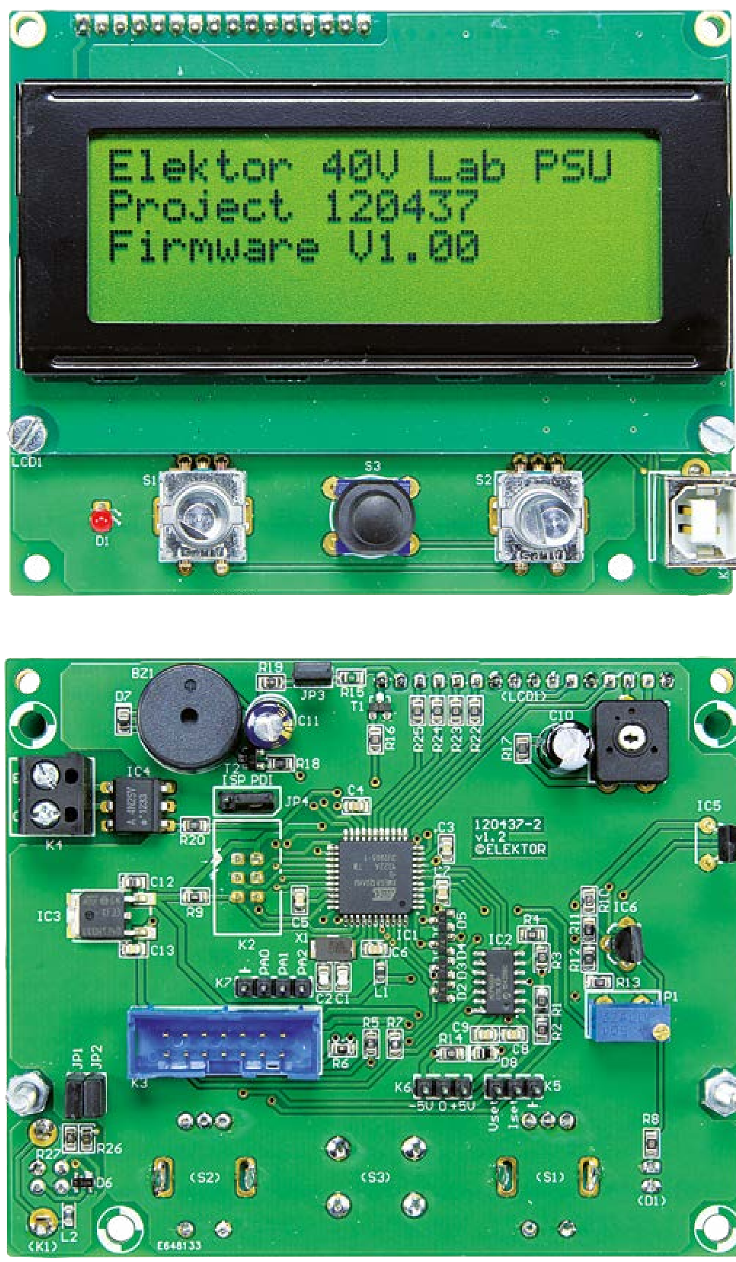


Figure 3.
Top and bottom sides of the fully assembled control board. All of the controls and indicators, including the LCD module, are located on the same side.

Figure 4.
The LCD module is secured to the PCB by a pair of M3 screws and nuts, with extra nuts for the right spacing.

the previous article (for the supply board) about mounting the SMDs also apply here: you need a certain amount of experience in order to solder these devices properly by hand. If you do not have sufficient experience, we recommend that you do not start on this project or that you get help from a more experienced electronics enthusiast or technician.

Mounting the components on the side with the encoders, pushbutton and display module is easier because they are all conventional components. Use short wire bridges or a 16-way pin header to mount the LCD module a few millimeters above the PCB, so that it cannot touch any protruding wires on the control board. Secure the LCD module to the PCB with M3 screws and nuts in the two holes at the bottom edge of the module. A pair of nuts between the module and the PCB provide exactly the right spacing (see **Figure 4**). Screws are not necessary at the top edge of the module, since the connecting wires or the pin header provide the right spacing and firm attachment.

Once all the components are mounted, you can fit jumpers at positions JP1 and JP2 for the USB port, JP3 for the LCD backlight, and JP4 for PDI programming mode. Now the board is ready for use with the supply board.

Use a short 14-lead flat cable to join connector K3 to connector K3 on the supply board. Keep this connection as short as possible to avoid interference. No other connections between the two boards are necessary. Actually you don't have to join the boards together just yet, since we plan to tell you how to mount the boards in a suitable enclosure in next month's issue of Elektor.

As usual, there are two options for programming the microcontroller. You can order a preprogrammed microcontroller from the Elektor Store (item number 120437-41; see [2]), or you can program the microcontroller yourself from Atmel Studio after you assemble the board. For the latter option you need a suitable programmer, such as the Atmel AVRISP mkII, connected to programming connector K2. As usual, the source code and hex files (120437-11) can be downloaded free of charge from the web page for this project [2].

Next month we will tell you more about the firmware, fitting the boards in an enclosure, and power supply calibration and adjustment.

(140373-I)

Web links

[1] www.elektor.com/120437

[2] www.elektor.com/140373

[3] www.elektor-labs.com/120437

Red Pitaya

Not just a USB scope module

By **Martin Ossmann**
(Germany)

Red Pitaya is the first open-source instrumentation platform that can be configured for various measurement tasks. Ready-made apps are available on the Web for downloading, but you can also program your own specific measurement processes. Several potential measuring tasks and demo programs are described in this article.

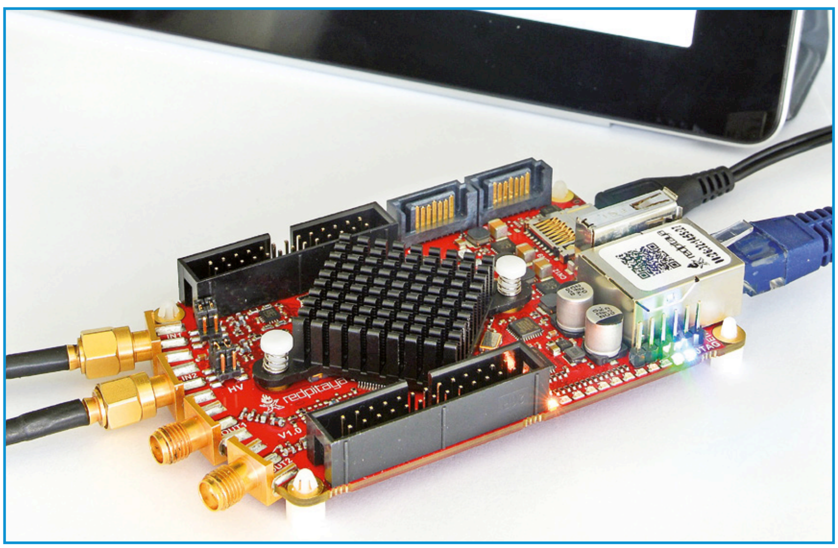


Figure 1.
The Red Pitaya module is very compact.

To get an idea of what the module (**Figure 1**) can do, have a look at the block diagram in **Figure 2**. The central component is a Xilinx ZC7Z010 system-on-chip (SoC) device. It contains a dual-core ARM9 processor clocked at 800 MHz, which hosts a Linux distribution on the Red Pitaya board. The SoC also contains an FPGA with 28k logic cells, a bunch of block RAM (BRAM), and several DSP slices. The FPGA can be used to implement functions, such as the functions used in digital signal processing, for extremely fast execution. The SoC is surrounded by a variety of peripheral devices and 512 MB of RAM.

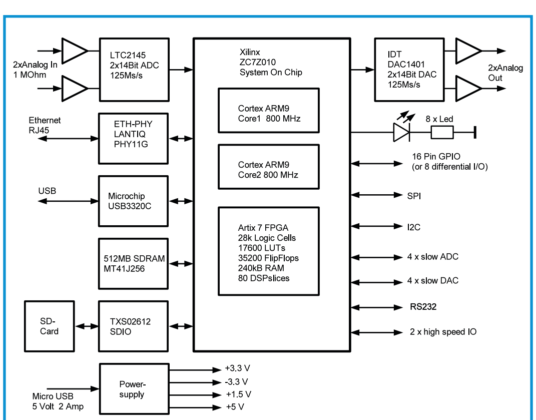


Figure 2.
Block diagram of the Red Pitaya module.

There are two 14-bit A/D and D/A converters for measurement tasks, with a maximum sample rate of 125 MS/s. With those specs you can handle signal conversion for fairly demanding tasks, and the FPGA gives you the necessary processing speed. The board features an Ethernet port, a USB Host port and a USB Com port for easy communication with a PC over a serial interface. There is also a slot for an SD card, which the Linux OS used for mass storage instead of a hard disk drive. General purpose I/O, I²C, RS232 and additional relatively slow A/D and D/A converters can be accessed through extension connectors. In short, the module has everything you need for demanding measurement tasks and high-performance signal processing with relatively high-frequency signals. That's exactly what Red Pitaya is supposed to be: a general-purpose instrumentation platform for signals up to 50 MHz.

Getting started

Getting started does not present any problems because a good user guide is available on the website. Simply copy all the software and data that is supposed to be on the SD card to a MicroSD card and insert it in the card slot on the Red Pitaya board. Then connect the Red Pitaya to your LAN and switch on the power. The module

boots up automatically, and you're ready to go. You might want to monitor the boot process on your PC via the USB Com port, since that way you can see which IP address is assigned to the Red Pitaya. After the module has booted up, you can communicate with Linux directly over the USB Com port or you can use Ethernet for further communication. The latter is the preferred method.

Instruments from the Bazaar

One of the main objectives of the Red Pitaya project is to provide ready-made applications in the form of apps available online. You can find the apps in the Red Pitaya Bazaar. After you point your web browser to the Red Pitaya website, you can go to the Bazaar from the home page and select the application you want. The apps listed in **Table 1** are currently available.

Of course, many other interesting apps for various measurement tasks are likely to show up in future. All of the projects are open source, so you can use them as starting points for developing your own applications.

We chose the oscilloscope and spectrum analyzer apps to get familiar with the device. We used them to make measurements on the circuit shown in schematic form in **Figure 3** and in physical form in **Figure 4**.

The inputs of the Red Pitaya A/D converters are buffered by opamps with 1 MΩ input impedance, so you can connect normal scope probes directly to the inputs. You can select a measuring range of 2 V_{pp} or 20 V_{pp} by setting the input gain jumper. If you run the oscilloscope app on the Red Pitaya and connect the probes to TP1 and TP2 of the circuit in Figure 3, you will see the screen shown in **Figure 5**. All of the usual oscilloscope settings are available.

When you run the spectrum analyzer app, the screen shown in **Figure 6** is displayed. You can track and compare the two spectra in real time. In Figure 6 you can easily see the harmonics of the crystal oscillator, with stronger harmonics on TP1 at the gate output than on TP2.

After getting an initial impression of what can be done with the Red Pitaya platform, we looked at what you need to know in order to program applications for this device (**Figure 7**).

Web-based instruments

The A/D and D/A converters usually interface directly with the FPGA. The FPGA is also used

Table 1. Apps presently available in the RP Bazaar

- Two-channel signal generator, 0–50 MHz
- Two-channel oscilloscope, 2x 125 MS/s
- Two-channel spectrum analyzer
- Frequency response monitor
- PID controller

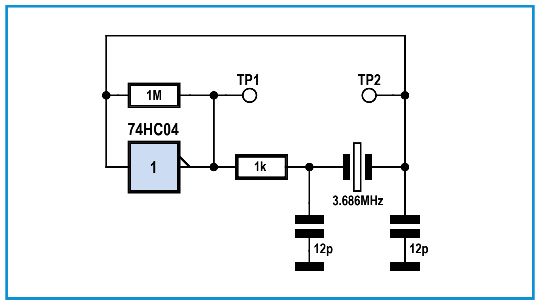


Figure 3.
A crystal oscillator circuit.

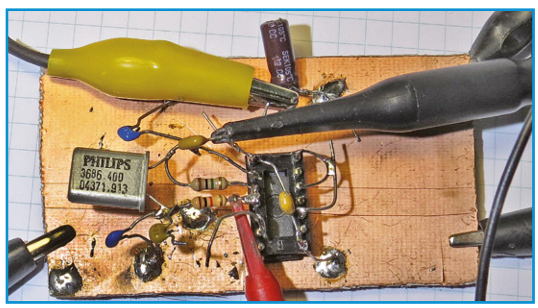


Figure 4.
Breadboard version of the crystal oscillator.

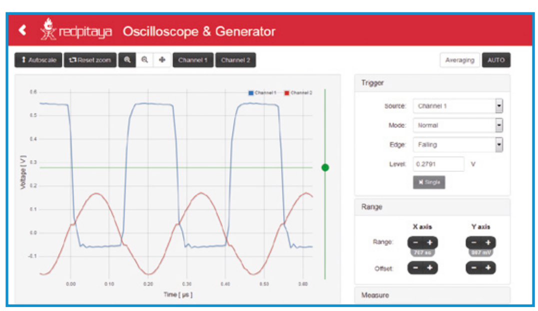


Figure 5.
The Red Pitaya scope app in action.

Figure 6.
The Red Pitaya spectrum analyzer app in action.

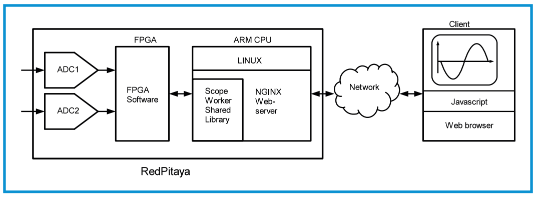


Figure 7.
The components involved in a Red Pitaya web-based instrumentation app.

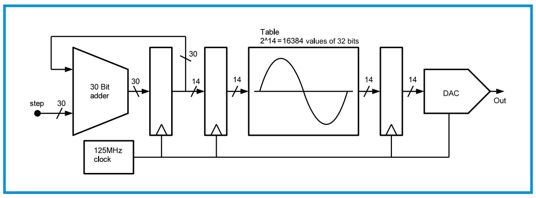


Figure 8.
Block diagram of the signal generator.

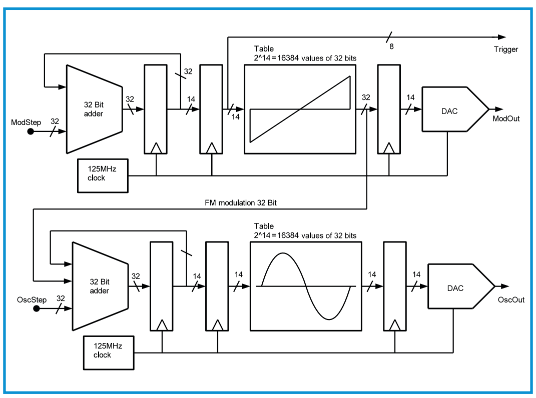


Figure 9.
Block diagram of an FM generator consisting of two DDS stages.

for any required pre-processing, such as storing frames in its fast internal memory. This means that you have to write suitable code to program the FPGA.

The FPGA usually communicates with a program running on the ARM9 processor. In most cases this program handles the control tasks, which do not require high processing speed but are more complex. Naturally, this code also has to be generated for any new application. In the case of a web-based app, this code is linked in on the Linux web server as a shared library. The web server handles data transfers between the web browser on the client system and the program running on the ARM processor. The graphical user interface (GUI) on the browser is implemented using a language such as JavaScript. Here again, this code must be developed for a new app.

As you can see, you need to know a good deal about programming if you want to develop complex instrumentation apps. However, you can take advantage of the fact that all existing apps for the Red Pitaya are open source, so there is plenty of starting material you can use to work your way toward a solution by making small changes. Now let's see how you might get started.

Programing the ARM processor

First let's look at what you can do with simple software for the ARM processor in combination with existing FPGA code. The standard FPGA code includes functions for a two-channel signal generator. The waveform for each channel is generated using a lookup table stored in memory with a total of 16,384 samples. Samples are taken from the lookup table incrementally in the same way as a direct digital synthesis (DDS) signal generator. These samples are fed to the 14-bit D/A converters. The sampling rate is 125 MHz. The block diagram for one channel is shown in **Figure 8**.

The configuration parameters are stored in FPGA registers, which can be read and written by the processor at specific addresses. This is done by a C program.

FM RTTY signal generator

Suppose you want to shift the frequency of the signal generator back and forth under software control. To do this, you change the step size of the DDS signal generator. The result is an FM RTTY signal generator. The wait time between each change is 20,000 µs, which corresponds to a data rate of 50 bits per second. The two transmit frequencies can be set to any desired value between 0 and 50 MHz. In the demo software the bits to be transmitted are first calculated as Baudot code and then output in a loop. The program code for this loop is shown in **Listing 1**.

You can directly modify the FPGA register by simply accessing the variable **`chb-count-step`**. This example shows that with a bit of simple of C programming, you can implement truly interesting applications with the Red Pitaya. Of course, you have to generate the C code using a complier that is compatible with the ARM9 CPU. For this purpose the author used the ARM9 GCC compiler running under Lubuntu on a virtual machine under VirtualBox.

Another option is to generate C code for the Red Pitaya under Windows, for which good descriptions are now available. In that case you can use SCP/WINSCP or another suitable utility to download the code from the PC to the Red Pitaya and then run the program.

FPGA programming

The next step is programming the FPGA, for which we also have demo software. The author used the Vivado tool from Xilinx, in the form of the free WebPACK edition, to generate the FPGA software. The standard FPGA code for the Red Pitaya is also available as a Vivado project, so you can use it as a starting point for your own designs by modifying the original code. In particular, that saves you the effort of reinventing the interface to the ARM processor and the other hardware. It is also helpful that FPGA register contents and signal samples can be viewed as variables in ARM processor memory. This enormously simplifies debugging because you can peek into the FPGA while it is running. Now let's look at an actual application.

FM signal generator

We already have a two-channel signal generator in the FPGA, with each channel arranged as shown in Figure 8. Now we want to modify this so that one channel modulates the frequency of the other channel. Since the waveform of the modulating signal is also stored in memory, you can use the device as a sweep generator by making the modulation linear.

Figure 9 shows the block diagram of our FM signal generator The upper portion is the modulator DDS. The samples in the table have a word size of 32 bits to enable smooth frequency modulation. The samples read from the table are sent to the second D/A converter for monitoring. Modulation is performed by sending the samples to the adder of the second DDS in the lower part of the figure, which forms a modulated DDS oscillator. To allow an oscilloscope to be triggered when the signal generator is used in sweep mode, a suitable trigger signal is tapped off from the upper DDS. It is routed to GPIO pins and to a software-readable register.

The essential software for the modulator shown in **Listing 2** consists of a few declarations and a code block with six instructions.
Figure 10 shows the spectrum of the output signal obtained by using this FM signal generator to produce a 10 MHz signal modulated by a 1 kHz sine wave with a specific modulation depth. The modulation depth is set to produce a frequency deviation of ±2 kHz on the 10 MHz carrier frequency.

The Verilog code of the frequency modulated oscillator is very similar to that of the modulator (see **Listing 3**). One of the main differences is in the computation of the new sample position, since the modulation signal data **fmInput-i** is also added here.

Listing 1. RTTY transmit loop.

```
uint32_t delay1=20000 ; // 50 Bit/sec
while(1){
  for(int k=0 ; k<nBits ; k++){
    int theBit=bitBuffer[k] ;
    if(theBit){ g_awg_reg->chb_count_step = step2 ; }
          else{ g_awg_reg->chb_count_step = step1 ; }
    usleep(delay1) ;
    }
  }
```

Listing 2. Verilog code of the DDS modulator.

```
reg [ 32-1: 0] dac_buf [0:(1<<14)-1] ; // data buffer
   16384x32 Bit
reg [ 32-1: 0] dac_rd    ; // DAC value
reg [ 14-1: 0] dac_rp    ; // DAC read pointer
reg [ 32-1: 0] dac_pnt   ; // read pointer

always @(posedge dac_clk_i) begin
   dac_rp   <= dac_pnt[14-1+18:0+18];
   dac_rd   <= dac_buf[dac_rp] ;     // read data value
   signal_o <= dac_rd ;
   dac_o <=dac_rd[14-1+12:0+12] ;    // feed to output
   dac_pnt <= dac_pnt + set_step_i ; // get new position
   trigSignal_o <= dac_rp[8-1+6:0+6] ;
end
```

Listing 3. Verilog code of the DDS oscillator.

```
reg  [ 14-1: 0] dac_buf [0:(1<<14)-1] ;
reg  [ 14-1: 0] dac_rd ;
reg  [ 14-1: 0] dac_rp ;
reg  [ 32-1: 0] dac_pnt ; // read pointer

always @(posedge dac_clk_i) begin
  dac_rp    <= dac_pnt[14-1+18:0+18];
  dac_rd    <= dac_buf[dac_rp] ;
  dac_o <= dac_rd[13:0] ;
  dac_pnt <= dac_pnt + set_step_i + fmInput_i ;
end
```

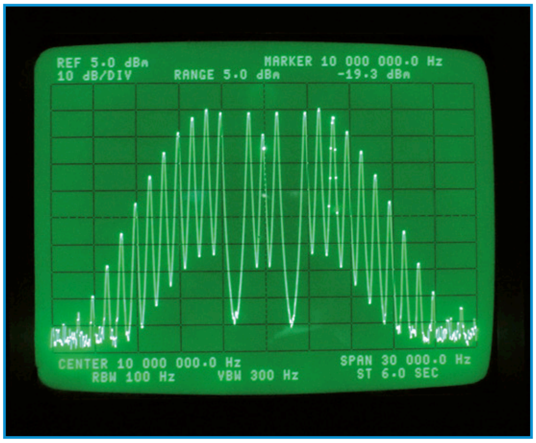


Figure 10.
FM signal spectrum.

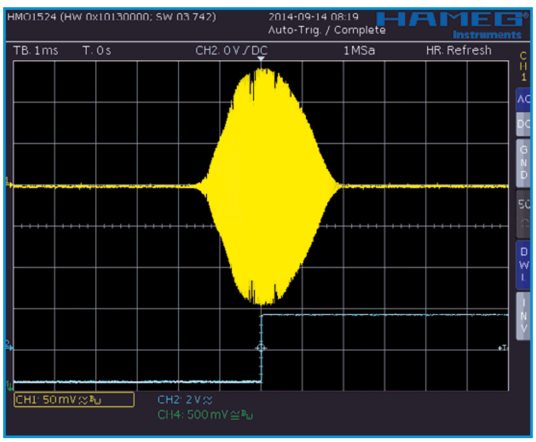


Figure 11.
Frequency sweep of a 10.7 MHz IF filter.

We configured the signal generator to sweep from 9.7 MHz to 11.7 MHz and used the output to measure the frequency response of a 10.7 MHz IF filter for an FM receiver. The result is shown in **Figure 11**. The blue curve at the bottom is the trigger signal.

Other projects

Even with simple FPGA and C programming, you can implement useful and sophisticated test and measurement devices with the Red Pitaya. To finish off, here are a few brief notes on other projects you can implement.

AM signal generator

Instead of the previously described FM signal generator, you can build an AM signal generator. The A/D converters can be used to modulate the oscillator waveform with external signals.

RMS voltmeter

The inputs are sampled at 125 MS/s, and the RMS values are computed and displayed. This gives you a two-channel RMS voltmeter for measuring signals up to 50 MHz.

Gain, phase and impedance analyzer

To measure the frequency and phase response of amplifiers, filters and other devices in the range from 1 kHz to 50 MHz, you generate a sine wave signal and then measure its amplitude and phase at the input and output. To determine the impedance, you measure the voltage and current and calculate the impedance from these values. This application is described in the forum area of the Red Pitaya website backyard, and the software is available from the Red Pitaya GitHub.

AM receiver for 0 to 50 MHz

The signal picked up by a wire antenna is fed directly to one of the A/D converters. It is mixed with an oscillator signal at the desired receive frequency (sample rate 125 MS/s). The quadrature components are low-pass filtered and reduced to 120 kS/s. These signals are sent by FIFO to the ARM processor for demodulation. The demodulated signal is output by one of the D/A converters. The result is a simple SDR AM receiver.

SSB signal generator

The speech signal is sampled by one of the A/D converters and converted into I and Q signals by a pair of filters in the ARM processor. These signals are sent to the FPGA at 120 kS/s. There they are interpolated and filtered, and then used to modulate the sine and cosine of the carrier signal. The result is an SSB transmitter module with a frequency range of 0 to 50 MHz.

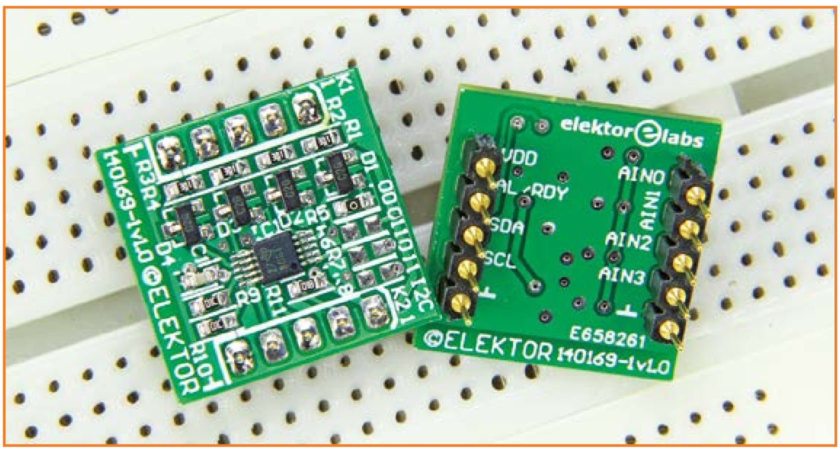
Outlook

It's obvious that the extremely high-performance hardware of the Red Pitaya module can be used for a lot of other interesting tasks. More and more devices will likely be developed as open-source projects in the course of time, which means that the module can easily pay for itself.

(140277-I)

ADS1115-eBoB

A drop-in 16-bit 4-channel converter on I²C



Design: **Ton Giesberts** (Elektor Labs)
Text: **Jan Buiting** (Editor-in-Chief)

As far as microcontrollers and some of their owners go the analog world out there first needs digitizing before it can be understood. This miniature but powerful ADC breakout board (BoB) does the job.

ADCs and their counterparts called DACs are friendly messengers running back and forth across the terrifying bridge between Analog Camp and Digital Camp. Although our experimental ELF Receiver and 16-bit Data Logger published in recent issues [1],[2] appeared vastly unrelated as far as subject matter and target audience are concerned, in fact they have one thing in common: the ADS1115 Analog-to-Digital Converter (ADC) chip from Texas Instruments [3]. It's perfect for designing a generic front end for relatively slow (< 1 kHz) but precise data acquisition systems (DAQs). Here we have the chip "BoB'ed'' especially for you.

In the 16-bit Datalogger article the ADS1115IDGST chip landed on a mixed-technology (SMD/TH) board no. 130485-1 in the company of an opamp, some connectors, and an LED driver. The lot was designed for easy connection to a whole range of Elektor embedded and related boards including the EEC/Gnublin series, Xmega Webserver, and the Extension Shield. The ELF Receiver is the odd man out, being a mostly analog application for hours of fun dredging in the noise.

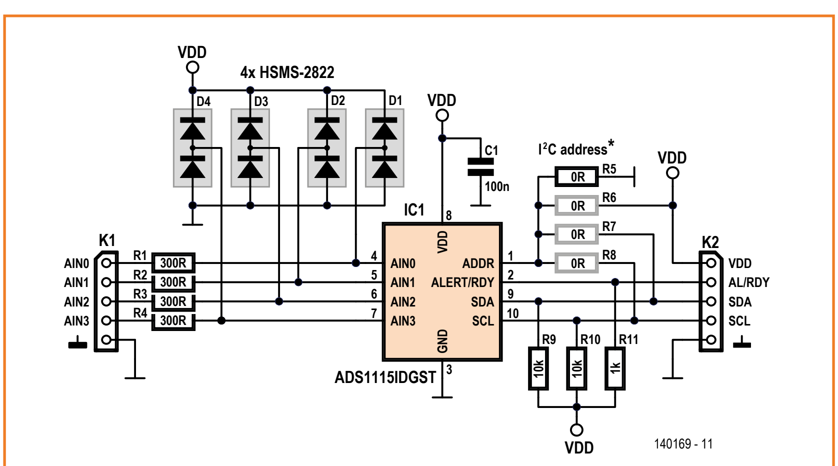


Figure 1.
The very essence of the ADS1115-BoB (Breakout Board) is its low component count.

The problem with them new ICs

is that they are supplied in packages you cannot easily solder by hand. Mainly triggered by the good response to the above two Elektor articles and pitying all of you dying to get their hands dirty with a *real fine chip* from 'Texas', it was decided to Elektorize the ADS1115IDGST little chip, meaning we've put it on a minimalist but consciously designed breakout board (BoB) for retailing through the Elektor Store (eBoB).

eBoB = bare bones

As shown in the schematic in **Figure 1** preciously little is required to make a 4-channel 16-bit ADC

with a maximum of flexibility—that's the crux of an eBoB. Here we have followed TI's recommendations in the datasheets for input protection using resistors R1–R4 and protection diodes D1–D4. The 300-Ω resistors limit the leakage current from each Schottky diode to a negligible value equivalent to ½ LSB.

The ADS1115 is an I²C device with a sub range of four slave addresses, hence the presence of 0-Ω configuration resistors (*aka* jumpers) R5–R8 on the ADDR pin. Only one of these resistors should be fitted, the default is R5. Information on the slave address selection is given in [2] and [3]. **Figure 2** shows the component mounting plan and the associated parts list of the ADS1115-eBoB, whose actual size is 19.1 x 19.7 mm. Although the ADS1115-eBoB comes ready assembled through the Elektor Store, some of you may have the tools and wherewithal to reproduce the project in the lab at even @ home.

Note that the labeled board pins are on two rows at the underside, allowing the ADS1115-eBoB to be plugged onto a breadboard. One row has the ADC inputs and ground, the other the power supply (VDD and ground), the I²C lines and the Alert/Ready line. If you do not foresee using the ADS1115-eBoB on a breadboard then regular 0.1" pitch pinheaders are fine.

Although this article is on hardware only and the convenience of a slow but highly accurate drop-in ADC on I²C we feel obliged to mention that [2] also provides some get-u-going code for Arduino and descriptions of linking to libraries. The lot is contained in an archive file previously released for the 16-Bit Data Logger project but also downloadable in connection with the present article [4].

(140169)

Web Links

[1] Extremely Low Frequency (ELF) Receiver, Elektor October 2014, www.elektor-magazine.com/140035

[2] 16-bit Data Logger, Elektor September 2014, www.elektor-magazine.com/130485

[3] ADS1115 docs: www.ti.com/lit/ds/symlink/ads1115.pdf

[4] Project software (identical to 130485-11.zip): www.elektor-magazine.com/140169

Component List

Resistors
R1-R4 = 300Ω 1% 0.1W, SMD 0603
R5-R8* = 0Ω 1%, 0.1W, SMD 0603
R9,R10 = 10kΩ 1%, 0.1W, SMD 0603
R11 = 1 kΩ 5%, 0.1W, SMD 0603
*Fit one only; default = R5

Capacitor
C1 = 100nF 10%, 16V, 0603 X7R

Semiconductors
D1-D4 = HSMS-2822-TR1G (Avago Industries)
IC1 = ADS1115IDGST (Texas Instruments)

Miscellaneous
K1,K2 = 5-pin pinheader, 0.1" pitch
Alternatively, for breadboard mounting:
K1,K2 = 5-pin pinheader, through hole, 0.1" pitch, round pins, Harwin D01-9923246
Ready-assembled board: Elektor Store # 140169-91
Optional: PCB only, Elektor Store # 140169-1, v1.0

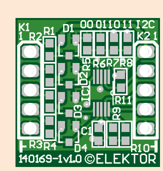


Figure 2. ADS1115-eBoB component overlay (actual size). In view of the SMD parts used, specifically the extremely small ADS1115, the board is available ready assembled.

Arduino Software Development with Atmel Studio

Debugging applications on the Arduino Due

By **Wolfram Pioch**
(Germany)

If you normally use Atmel Studio to develop software for microcontrollers in the AT(X)mega family and are familiar with the advantages of working with a debugger, you will sorely miss the convenience of these tools when you switch to the Arduino IDE to develop sketches for the Arduino Due board. This article describes a suitable remedy.

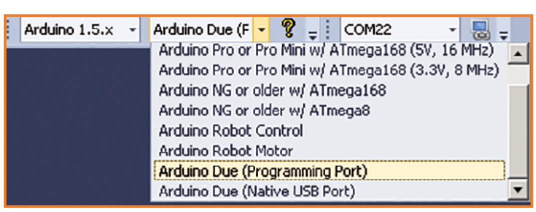


Figure 1.
The actual Arduino version, board and programming interface are selected on the toolbar.

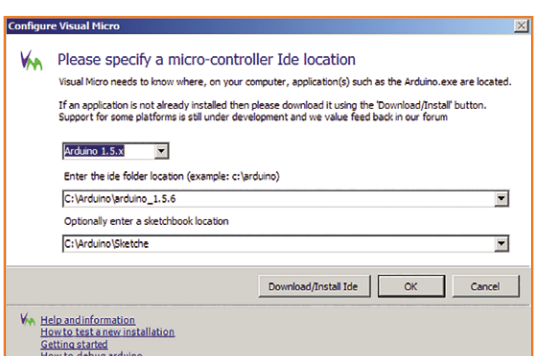


Figure 2.
The location of the Arduino IDE folder can be entered in the configuration window.

Although it's nice that you can get useful results quickly with the many ready-to-run Arduino sample sketches already available, when you start developing serious software for an Arduino board with an ARM processor you are confronted with the following significant shortcomings:

- Compiling the program in the Arduino IDE takes too long.
- Loading the program takes too long.
- No genuine hardware debugger is available.

To remedy this situation, the author has devised a suitable alternative.

One reason for switching to a 32-bit microcontroller is faster access to the hardware, especially the I/O pins. Accordingly, a good way to test the processing power of a microcontroller is to write a sample program (Arduino sketch) that gener-

ates a pulse waveform with the shortest possible pulse width and measure the pulse width of the output signal. In this article we describe how you can program a test sketch for this purpose using Atmel Studio 6.2 [1].

Arduino compatibility

The procedure described here assumes you have Atmel Studio 6.2 (or later) installed under Windows. To make the Atmel IDE compatible with Arduino, you also have to install a free add-in called Visual Micro [2], which is an Arduino IDE for Microsoft's Visual Studio and Atmel Studio. Visual Micro is directly available in Atmel Studio after it has been installed.

The USB-based Arduino debugger available from [2], which is not free, is not necessary because debugging with the Atmel ICE is significantly more convenient. The latter tool is available in various webstores at prices in the 100 dollar range (see the review in the October 2014 issue of Elektor [3]). The expenditure is worthwhile if you use Atmel microcontrollers more than just occasionally.
With this debugger you can set real breakpoints without recompiling the program, observe variables in the Watch window, and examine or change memory contents. What's more, you can inspect the numerous I/O registers and alter them with a mouse click.
After you install Visual Micro, a new toolbar appears in Atmel Studio. On this toolbar you can select the current Arduino version (1.5.x), the board (Arduino Due) and the programming interface (programming port) as illustrated in **Figure 1**. After this you have to configure the appropriate virtual serial interface, which you can find by looking in the Device Manager window when the Arduino Due is connected to the PC with a USB cable. As in the Arduino IDE, the serial monitor button is located to the right of these settings.
If the settings do not appear automatically, for example because the Arduino IDE software is not located in the default directory, you have to select "Configuration Manager". That opens the window shown in **Figure 2**. There you have to enter the correct folder location manually, since no selection dialog is available.
Clicking the question mark on the menu bar in Figure 1 opens the Micro Explorer window (**Figure 3**), which is the counterpart of the Solution Explorer window in Atmel Studio. There the Arduino references are shown as links on the Reference tab. A click on one of the entries takes you to the corresponding website in the configured browser. The Examples tab holds a list of examples (including library examples) from the Arduino IDE, grouped by topic.

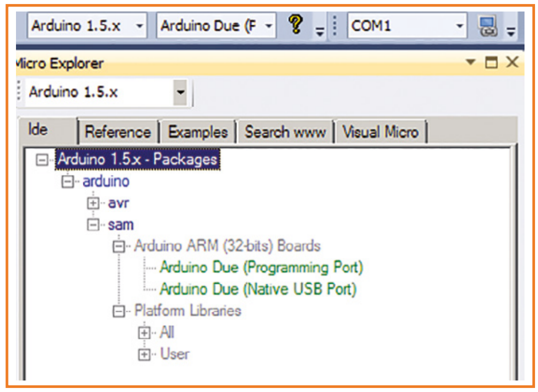


Figure 3.
The Micro Explorer window of Visual Micro.

Sketches in Atmel Studio

A major benefit of the Atmel Studio IDE is the code completion function, which you now have available for Arduino sketches as well. For this you have to enable Visual Assist X via *VAssistX → Enable/Disable*. With this enabled, all possible completion options are listed each time you type a character in the Code window. For example, if you type "S" at the start of a new line in a sketch, the terms Serial, Server, SPI and so on are suggested. You can then select the appropriate term directly, with no risk of typos. Even better, after you type a dot, for example after the term "Serial", the existing attributes and methods of this class are listed for selection (**Figure 4**).

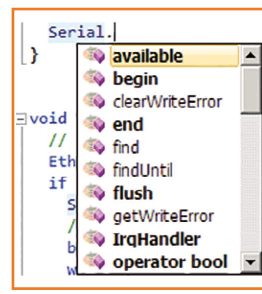


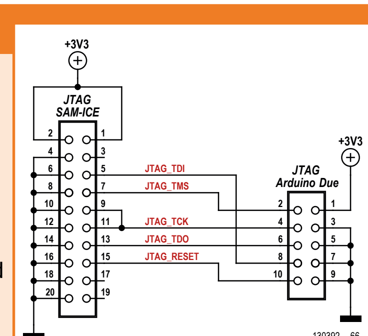
Figure 4.
The Atmel Studio editor helps you avoid typos by suggesting function names and options for autocompletion.

At this point you could create or open a sketch in the Arduino IDE in the usual manner and then import it into Atmel Studio. However, you now

Adapter for SAM ICE

This adapter is necessary for using the SAM ICE debugger/programmer with the Arduino Due.

It connects the 10-pin JTAG pinheader with 50 mil (0.05"; 1.27 mm) pin pitch on the Due board to the 20-pin JTAG connector on the SAM ICE device.



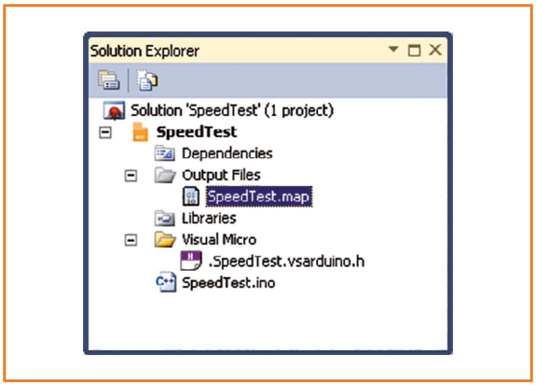


Figure 5.
The Solution Explorer window of Atmel Studio.

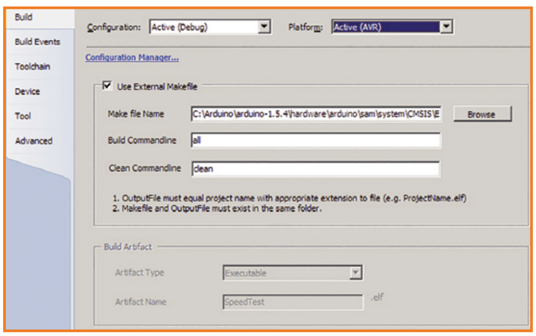


Figure 6.
The wrong platform is shown here, but everything still works fine.

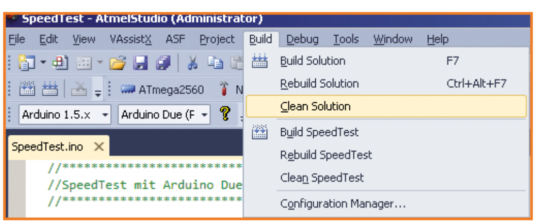


Figure 7.
Compiler run with the Clean Solution option.

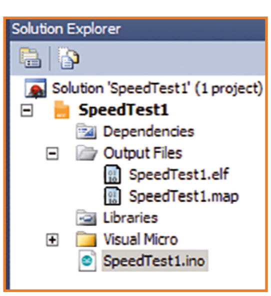


Figure 8.
Output files in the Solution Explorer window.

have the option of creating new Arduino sketches directly in Atmel Studio. After you create and save the SpeedTest sketch (see **inset**) in either of these two ways, the structure shown in **Figure 5** appears in the Solution Explorer window. Now you can open the sketch and edit it in the customary manner in Atmel Studio. From this point on you don't actually need to run the Arduino IDE. However, you should not delete it because it has to remain available on the PC.

It's a bit confusing that the platform is shown as "Active AVR" in the project properties (right-click SpeedTest: *Properties* → *Build*), at least in the current version of Atmel Studio (6.2). Unfortunately there's nothing you can do about this. The platform is also shown as "AVR" under "Toolchain" and "Device". Nevertheless, everything works as it should. When you select "Arduino Due" as the board, the project is compiled in the same way as with the SAM platform for Atmel ARM microcontrollers. According to **Figure 6** the AVR platform is active, but the "Make file Name" location is correct for the SAM toolchain of the Arduino IDE. As usual with Atmel Studio, pressing the F7 key compiles the project and pressing the F5 key (or clicking the solid green arrow) downloads the program to the board via the boot loader. You have to select "Clean Solution" on the Build tab (**Figure 7**), since that is the only way to ensure that all files are compiled the same way as in the Arduino IDE.

As previously mentioned, one of the main advantages of Atmel Studio over a pure Arduino IDE is shorter compilation time, which significantly accelerates the iterative process of compiling, testing and modifying program code. That's because the Arduino IDE always compiles all of the program files, while Atmel Studio only compiles the files that have been modified. In most cases this means that only the `.ino` file, which contains the sketch code, is compiled. On the author's PC the first compiler run for the SpeedTest sketch took 3.7 seconds, but the second run after a change was finished in just 0.23 second.

Debugging with the Atmel ICE

A real hardware debugger such as the Atmel ICE makes software development a lot easier. All you need to connect the Atmel ICE to the Arduino Due is the basic cable that comes with the device. The Arduino Due can be powered from the USB port or by an external AC adapter. The Atmel ICE only needs the USB port.

There are two options for plugging in the debug cable. The SAM port is the right choice for the Arduino Due, with the other end of the cable plugged into the JTAG connector on the board.

The current version of Atmel Studio (6.2) does not support editing of Arduino sketches on an Atmel ARM platform, but there is a way to get around this. To use the Atmel ICE for debugging, simply launch a second instance of Atmel Studio. Then select *Open -> Open Object File For Debugging*. The compiled files generated by Atmel Studio are located in the following folder under Windows 7:

```
C:\Users\xxxx\AppData\Local\
VMicro\Arduino\Builds\SpeedTest\
arduino_due_x_dbg\
```

There you will find the file **`SpeedTest.elf`**. Now let's open it for debugging.

If the sketch was compiled and saved with the

Arduino IDE, a copy of this file is also located in the debug folder for the sketch. It can be seen in the Solution Explorer window of the code generation instance of Atmel Studio under "Output Files" (**Figure 8**).

Using the browse function, find and select the output file **`SpeedTest.elf`**. You should see a window like the one shown in **Figure 9**. You can specify your own project name and storage location for the debug project folder. Your settings will be saved and used automatically the next time you open the debug project.

Next you see a window for selecting the device family and target microcontroller (**Figure 10**). For the Arduino Due you should select "SAM3, 32 bit" for the device family and "ATSAM3X8E" for the microcontroller type.

Now the file **`SpeedTest.ccp`** appears in the Solution Explorer window. This is simply the program file of the sketch, which originally had the extension **`.ino`**. Right-click the project name "SpeedTest_Debug" to open the Properties window.

When the Atmel ICE is connected to the PC over USB, the debugger can be selected under *Tool -> Selected debugger/programmer.* Here you must specify "SWD" as the interface instead of "JTAG".

Now the Atmel ICE is ready. You can also use it as a programmer. To open the programming window, press *Ctrl-Shift-P* or click the chip symbol with the lightning flash. The desired functions are available after you click *Apply*. You can set the maximum SWD clock rate (2 MHz) on the Interface Settings tab. On the Memory tab, you can download the **`SpeedText.elf`** file directly to the flash memory of the microcontroller. The program starts running automatically after the programming process completes.

However, it is more efficient to use the *Debug → Start Debugging and Break* or the *Start Without Debugging* command (or the corresponding icon: the blue arrow with the vertical blue bar or the solid green arrow) to download a program for debugging or running, instead of the device programming function.
Downloading the complete program to the microcontroller takes roughly 1 second with the Atmel ICE, which is quite fast.

The actual structure of the downloaded Arduino program can be seen in the Solution Explorer window, where the **`main.ccp`** file is selected and the cursor is positioned on the first line of this routine

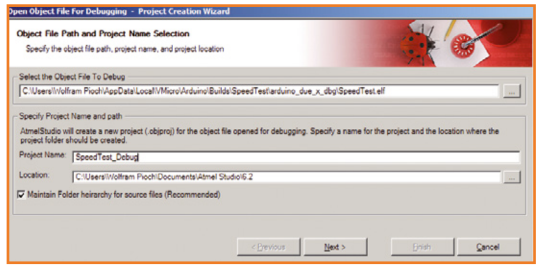


Figure 9.
Dialog for selecting the generated output file.

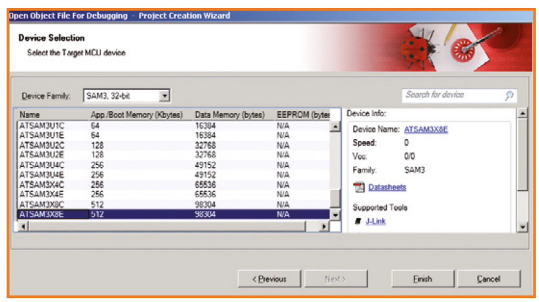


Figure 10.
Dialog for selecting the device family and device type.

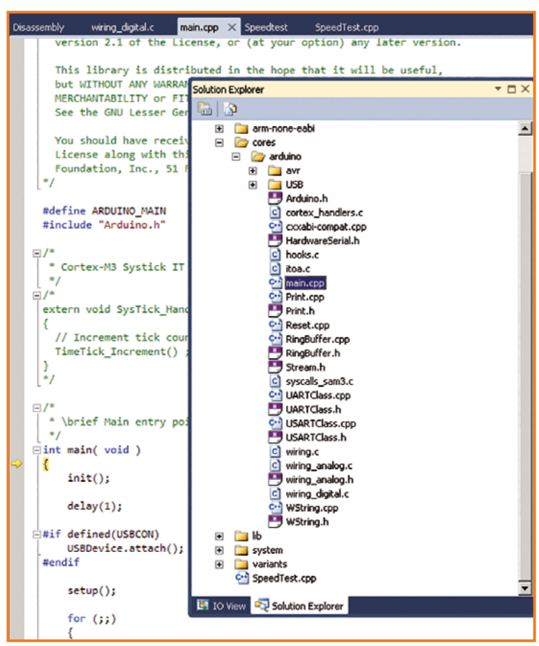


Figure 11.
The true complexity of an Arduino project can be seen in the Solution Explorer window.

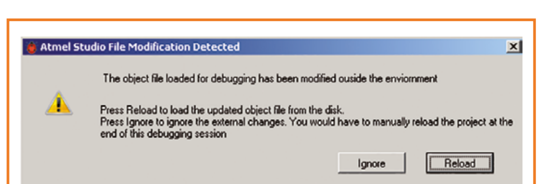


Figure 12.
When this error message appears after you modify a file, simply click "Reload" to dismiss the message.

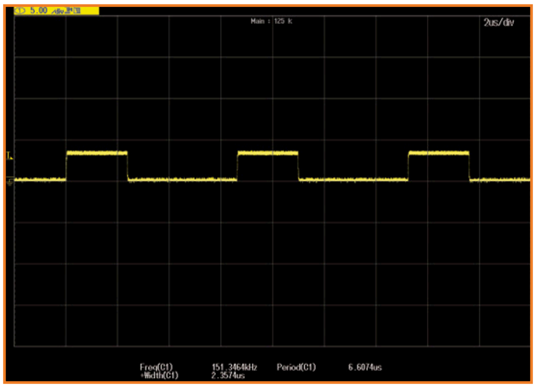


Figure 13.
Oscilloscope screenshot of the 150-kHz pulse waveform.

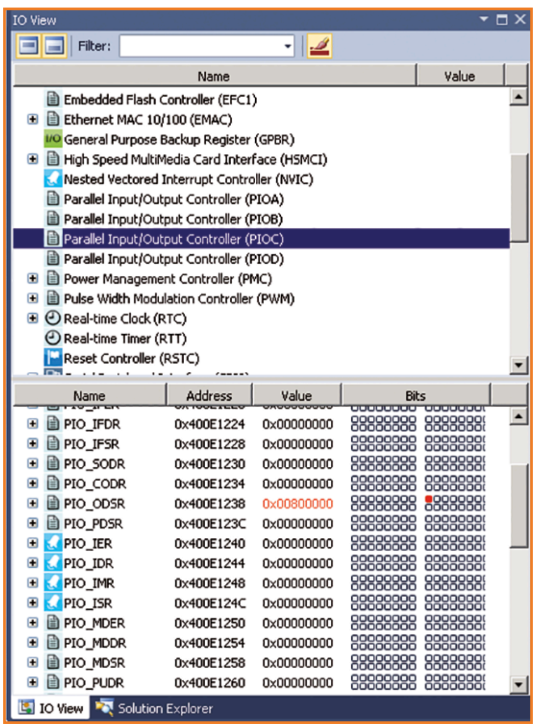


Figure 14.
Microcontroller PIO register.

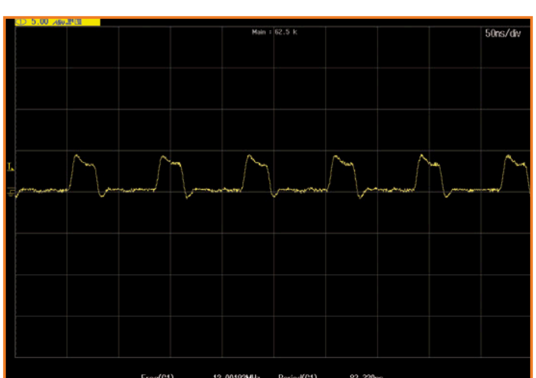


Figure 15.
Oscilloscope screenshot of the 12-MHz pulse waveform generated by version 2.0 of the program.

(**Figure 11**). As you can see, the `SpeedTest.ccp` file is only a small part of the overall program, which is why a full compilation takes so long. The debugger gives you access to all the debug functions of Atmel Studio, such as single-stepping with F11, setting breakpoints with F9, defining breakpoint conditions in the Breakpoint window, and so on.
If you edit and recompile the `SpeedTest.ino` file in the first instance of Atmel Studio, the message shown in **Figure 12** appears in the second instance (the debug environment). Press "Reload" to close the debug message and load the modified program into the microcontroller. Now the cursor is again positioned on the first line of the program code and the program can be run again.

If you have previously worked with 8-bit AVR microcontrollers in the Atmel Studio environment, you will not have any problems working with 32-bit ARM microcontrollers because the user interface is nearly the same. Incidentally, clearing and rewriting the flash memory with the Atmel ICE does not affect the boot loader because it is located in the ROM of the microcontroller and therefore cannot be deleted or overwritten. Consequently simply downloading a program over the USB interface still works the same way.

SpeedTest

Only the following two instructions are executed in the main loop of the example sketch (see the **SpeedTest listing**):

```
digitalWrite(TP1,1);   //On
digitalWrite(TP1,0);   //Off
```

With an oscilloscope, you can see a pulse waveform on this pin while the program is running, with a pulsewidth of 2.35 µs and a period of 6.6 µs (**Figure 13**), which corresponds to a frequency of about 150 kHz. That is not very high, and you could hardly be blamed for thinking that there's no reason to use an ARM controller clocked at 84 MHz for this when a simple 8-bit AVR microcontroller can do it better.

The reason for this poor performance is that the Arduino instruction `digitalWrite` is a cover name for a bunch of C routines, as you can see by executing the program in single-step mode with the debugger (F11 key). To actually achieve high processing speed, you have to avoid this over-

head. Fortunately, the Arduino programming language is not a separate language, but instead a form of C with a GNU compiler in a special Arduino configuration.
This means that a pin configured as an output can be addressed directly by setting or clearing a bit in the PIO register. The instruction

```
const int TP1 = 7;  //Test pin
```

selects Arduino pin 7 as the test pin. According to the Arduino pin mapping, this corresponds to pin PC23 of the microcontroller or bit 23 in the parallel input/output (PIO) register. With the program stopped you can go to the I/O View window of the debugger (**Figure 14**) and click bit 23 of the PIO_ODSR (set) register to set the pin high. To reset the pin, click the same bit in the PIO_CODR (clear) register. In the main loop you can simply use the instruction

```
PIOC->PIO_SODR = 1<<23;
```

to set the pin without any fuss or bother.

To test this, and at the same time check whether conditional compilation works properly, the author generated a new version of the sketch (see the **SpeedTest 2.0 listing**). If the expression `Direct` is defined in the code, the code segment after `#ifdef Direct` is compiled, and otherwise the previous version with the slow `digitalWrite` instruction is compiled. Naturally, this change would also work in the normal Arduino IDE.

Listing 1. Speed-Test.

```
//*******************************
//Speed-Test with Arduino Due
//*******************************

const  int TP1  = 7;  //Testpin

void setup()
{
  /* add setup code here */
  // set the digital pin as output:
  pinMode(TP1, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:

  digitalWrite(TP1,1);  //On
  digitalWrite(TP1,0);  //Aus
}
```

Listing 2. Speed-Test 2.0.

```
//********************************
//Speed-Test 2.0 with Arduino Due
//********************************
#include „arduino.h“
//
//
const int LED1 = 13;
      int LED2 = 12;
      int LED3 = 11;
      int TP1  = 7;  //Testpin

#define Direct

void setup()
{
  /* add setup code here */
  // set the digital pin as output:
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(TP1, OUTPUT);
  // set output low
  digitalWrite(LED1,0);
  digitalWrite(LED2,0);
  digitalWrite(LED3,0);
  digitalWrite(TP1,0);
}

void loop() {
  // put your main code here, to run repeatedly:
#ifdef Direct
 //x= state of PIO_SODR with bit 23 = 1
 int x = PIOC->PIO_SODR | 1<<23;

 while (1){  //Only for testpurposes, don‘t exit this loop
  PIOC->PIO_SODR = x;
  PIOC->PIO_CODR = x;
 }

#else
  digitalWrite(TP1,1);
  digitalWrite(TP1,0);
#endif
}
```

The result of this code optimization, as shown in **Figure 15**, is stunning: the pulse waveform now has a pulse width of 23.2 ns and a period of 83.3 ns. This corresponds to 12 MHz – about 80 times faster than before. From that we can conclude that using an ARM microcontroller isn't such a bad idea.

Summary

Working with Atmel Studio accelerates Arduino sketch development to a completely new level thanks to shorter turn-around times, since only modified code is recompiled. Atmel Studio and the tools that can be used with it, such as the Atmel ICE as a debugger and programmer, are worthwhile when you work with Atmel microcontrollers because they offer professional debugging features and because downloading programs to the microcontroller memory is significantly faster.

Incidentally, you can also use Atmel's SAM ICE instead of the Atmel ICE for ARM projects. With the SAM ICE hardware debugger and programmer, microcontroller programming speed can be boosted by a factor of 4 because it can operate at 8 MHz instead of 2 MHz. For this you need the adapter described in the **inset**. If you are a professional software developer, the expenditure quickly pays for itself even if the device can only be used with the AT91xx microcontroller family.

(130392-I)

Web Links

[1] Atmel Studio 6.2:
www.atmel.com/microsite/atmel_studio6

[2] Arduino IDE for Visual Studio:
www.visualmicro.com

[3] Atmel ICE:
www.atmel.com/tools/atatmel-ice.aspx
Review: www.elektor-magazine.com/140275

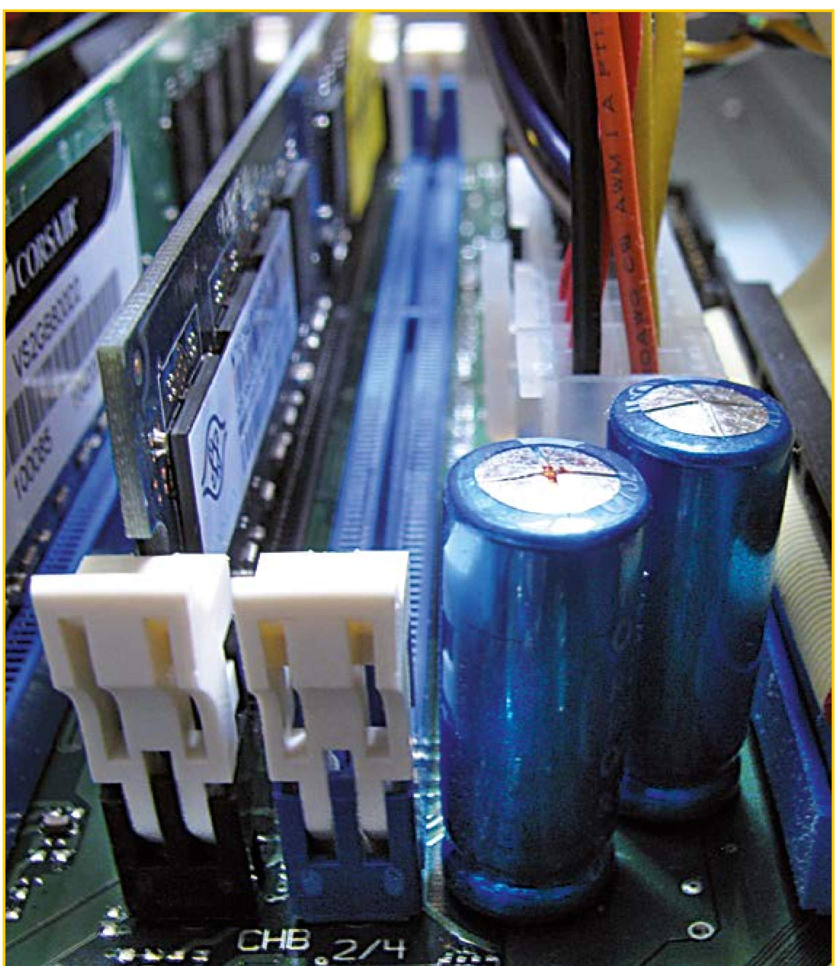
[4] SAM ICE:
www.atmel.com/tools/atmelsam-ice.aspx

Bulging Caps

By **Thijs Beckers**
(Elektor Labs)

The other day I went to the local Used Chemicals Drop-off to dispose of some sad old inkjet printers when a fairly intact looking PC case caught my eye. Without needing to *dumpster dive* I was able to spot an “Intel Core™ 2 Duo” sticker and another with the Windows Vista™ Home Premium OEM product key, these gave me a clue about the age of this PC. It wasn’t beyond hope considering that my current laptop was “Built for Windows Vista”—I´m reluctant to throw away stuff that still works, even though it might not be shiny and brand new anymore. Anyway, the PC case didn’t look battered or kicked around much, so

I decided to take it home and to find out about the specs—CPU, RAM, HDD, that sort of thing.



Back home I opened it up. All components were present except for the hard drive. And it looked like the previous owner had removed it decently and hadn’t just ripped it out. So I guessed it was time to try to fire it up.

I connected a monitor and hooked up a keyboard and mouse. I connected the power cord and pushed the Power switch on the front and... it came alive! I entered the BIOS to see more detailed specs and was happily surprised. Here’s a breakdown of my find:

- CPU: Intel Core 2 Duo E4500 @ 2.2 GHz
- RAM: 2 GB DDR2 @ 667 MHz
- Mainboard: Fujitsu D2151-A21
- Graphics: NVidia Geforce 8400GS 256 MB
- Optics: DVD-ROM and DVD-R/RW DL drives
- 350-W PSU
- Internal Card Reader
- Cooler Master case

In $2^{n>3}$ months we will have a good laugh about this configuration, but these days it’s good enough for casual surfing, mailing and even some music making. Some more dumpster-diving in certain places is sure to yield a hard drive and I can get this baby running.

But first there was an issue I didn’t like very much. I noticed it when I inspected the mainboard. See those capacitors next to the ATX connector? Though the mainboard functioned perfectly, my electrical engineering hunch says them caps look like they’re about to explode. Which is of course a bad thing. But if replacing a couple of bulging capacitors is the only thing needed to give this PC a second lease of life, I’m still a happy camper. I ordered a couple of decent low-ESR type replacement caps (Panasonic FM-series), removed the time bombs and soldered the new caps in.

After installing a fresh operating system, the PC ran fine. And after a month or so it still does. Did I mention I made a Hackintosh out of it? Maybe I shouldn’t have said that out loud ☺.

(140363)

Elektor.Labs goes wearable

By **Clemens Valens**
(Elektor.Labs)

Wearable is the new hype

The previous months have seen a lot of hobbyist-related activity from Intel, the world's largest manufacturer of PC microprocessors. Not only did Big Blue release their new Galileo (V2 is now available) and Edison boards with Arduino support, the company was also the main sponsor of the above mentioned Rome Maker Faire. At this show Intel made another announcement, besides the Edison board, that didn't really get the attention that it deserves, in my humble opinion: Intel firmly intends to become a major player in the still pretty small wearable electronics market. To prove this they showed of their MICA (My Intelligent Communication Accessory) bracelet.

Intel has invested in start-ups and teamed up with fashion designers to come up with reference designs (and products, why not?) to open the market and to show what's possible besides displaying the user's heart rate and body temperature. Have a look at the Mimo Baby Monitor, built on Edison.

Elektor cannot stay behind, of course, and so here's a call for wearable electronics projects to be published on Elektor.Labs. Have you designed something wearable? Do you have an idea for such a device? Post your wearable ideas and projects on Elektor.Labs and we will see that the best ones will be transformed into real products.

www.elektor-labs.com/wearable

3D printing—the story continues

Some months ago in this column I ranted about the useless things produced by so many people with 3D printers. I ended the article by suggesting that those people should print their own 3D printer filament or "ink" to save the planet from drowning in polymer 3D printed garbage. At the Maker Faire that was held at the beginning of October 2014 in Rome and that I visited I encountered once more many useless 3D printed objects. To be honest there were some very useful 3D printer applications on display too, sadly they were a small minority.

During the keynote presentation the winners of a product design contest organized by the Italian magazine *Focus* got announced. To my big surprise the winning project was a… 3D Printer Filament Extruder! A kind of 3D printer that prints 3D printer ink. Being an Italian project, great attention had been given to its looks, and the result is an appliance that sure looks great in any kitchen (unlike the ugly 3D printer). At the top you feed in polymer ingredients and out comes a filament that gets collected on a detachable spool. Very likely it's suitable for making spaghetti too.

Walking the show grounds I came across a few other similar machines, but not so nicely shaped as the EWE Filament Extruder. Photos are available on this webpage:

www.elektor-labs.com/3d-printing

(140365)

DesignSpark Tips & Tricks

Day #16: Working with Components

By **Neil Gruending** (Canada)

We've talked a bit about components in the previous installments so today let's look at them in more detail.

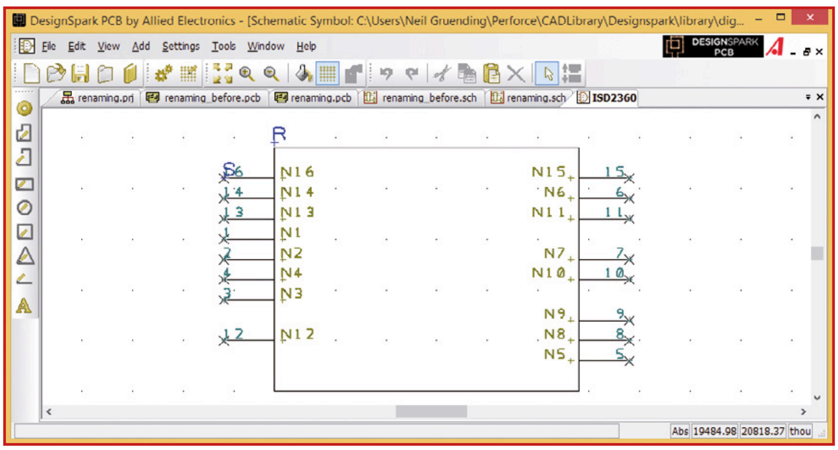


Figure 1.
Modified schematic wizard symbol.

Today I realized that we've talked about components in DesignSpark before without discussing them in detail or how to use them efficiently, so that will be our topic for today. Back in DesignSpark Tips & Tricks Day #3 (Elektor September 2013) I talked about libraries and how components in DesignSpark linked all of the design information for an electrical component in one place. That's how the schematic and PCB tools can find the component information they need without interfering with each other.

Creating and editing components

Creating and editing components can be a lot of work if you draw them from a blank page every time. If you can find a similar component that's already in the library then it's easiest to copy it using the "Copy To" button in the Library Manager window and then modify it. But when you can't, then the DesignSpark component wizards are a great way to make new components quickly. Let's build an audio playback IC component to see how the wizards work.

I always like to start with the schematic symbol first so we will open the DesignSpark schematic wizard by clicking on the Wizard button in the Library manager. Answer a few questions and it will automatically create simple schematic symbol that we can modify. After a little editing I came up with the schematic symbol in **Figure 1**. The pins are where I want them but if you look closely you can see that they have the default names N1 to N16 which is okay for now because they will be changed later at the component level. One useful trick that I used is to place temporary text strings over the pin names to make sure that the proper pin names will fit inside the symbol. DesignSpark also has a PCB footprint wizard that can create a wide variety of PCB footprints for you. The best part about it is that it can create all of the pads for you along with the correct spacings. The last step is to use the component wizard to associate the schematic symbol to the PCB footprint. Once you choose the schematic symbol and PCB footprint you've just created you will get to the Assign Pins screen like in **Figure 2** which is where you can label each pin name and associate the schematic symbol pins to the PCB footprint. In my case I designed the schematic symbol so that its pin numbers match the PCB footprint pin numbers so I would click the "Assign 1-to-1" button. But DesignSpark would let you have any pin mapping you want. This is also where you can enter all of the pin names in the Terminal Name column. Clicking on the Next button will create the component for you and let you save it to your library. **Figure 3** shows what our final component would look like in a schematic. But what if we needed to make changes and update our design? We will cover that next.

Updating components

When you make changes to a component in a library DesignSpark will only update the library files and not the schematic and PCB files that use it. That's because every design file keeps its own copy of the component so that changes in a component library can't accidentally affect an existing design. So how do we update those local component copies with the new component library information? One way is to delete a component and then add it back into the design but this is a dangerous

operation that could have unintended side effects like breaking net connectivity in your design. This would be especially risky in a PCB file because it could break the link to the schematic for that file. Instead you always want to go into the Tools menu and select “Update Components” and then choose which components you want to update. This works in schematic and PCB files.

You will almost always want to either browse for the components or choose to update the selected components. You don’t usually want to update all components automatically unless you review all of the pending changes to make sure that there won’t be any unexpected updates. Either way you will get the Update Components window in **Figure 4** if you update components in a schematic. The PCB version of the Update Components windows is very similar.

The “Only update item if version is different in library” option tells DesignSpark to only update a component if it’s different from what’s in the library. The “Keep value positions” option tells DesignSpark to try and leave all of visible component fields in their current locations. This usually works but you will still want to double check that nothing has moved on you after the update because even though DesignSpark will keep the same anchor point sometimes the text justification will revert to what was used in the library. Enabling the “Keep existing component values” option will keep all of the component value fields in your design, otherwise the values will be overwritten with the values in the library. This option is important when you modify a component’s value fields in the library and you want to transfer those changes to your design. The Update Components window adds another option called “Remove pad style exceptions” when you update a PCB component. You would normally leave it selected so that DesignSpark will clear all of the errors it detected with the previous footprint.

Replacing and changing components

Sometimes you need to replace or change a component as you work on a design. For example, I like to have a different resistor component for each value in a design so that I can put all of the manufacturer’s part numbers in the component value fields and make a proper BOM later. That means that I just change the component when I want to change a resistor’s value.

If you haven’t already placed the component on the PCB then it doesn’t matter if you just delete a component and then add a new one to replace it. But if you do this and the part is on the PCB, DesignSpark will delete the component and then add the new one to the component bin. A better solution is to replace a component without deleting it first. You do this by selecting the component and open its properties. Then you click on the Change button which opens the Change Component window where you can select the new component. With this method DesignSpark will leave the component on the PCB and you also don’t have to worry about matching the reference designator.

Conclusion

Today we’ve taken a pretty close look at how to create components with the DesignSpark wizards and how to update designs with the library changes. Next time we’ll learn how to use multiple component gates and hidden power pin connections.

(140367)

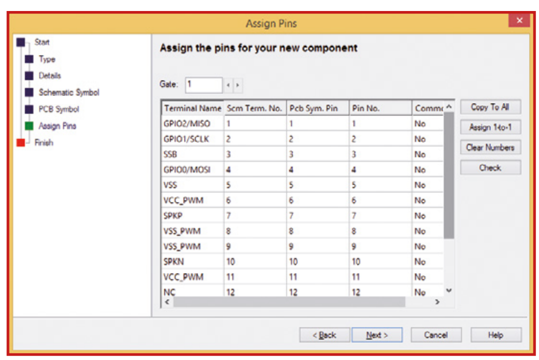


Figure 2.
Assign Pins screen.

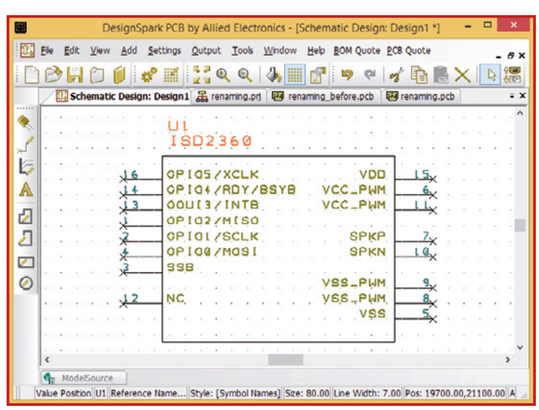


Figure 3.
Completed component.

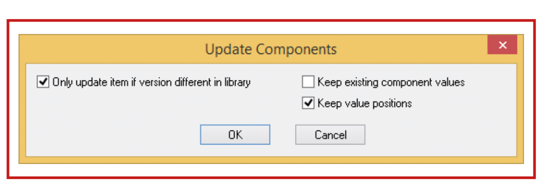


Figure 4.
Update Components window.

Tantalum Bead Capacitors

Weird Component # 10

By **Neil Gruending** (Canada)

1

Source: Wikipedia

2

What's the first thing you check when a piece of electronics stops working? I bet it's the power supply voltages and if you are repairing a board that over 20 years old then you've probably your share of tantalum bead capacitor failures. If you're lucky they will just short out and current limit the power supply but I've also seen them explode into a small ball of flame! But whence the fail… the noise… the vile smell?

Tantalum capacitors like the dipped ones in **Figure 1** are a form of electrolytic capacitor that uses oxidized tantalum powder pressed into a pellet as its dielectric. The tantalum powder allows capacitors to be built with a very high capacitance for their size and since the powder is solid they also have a very wide temperature range. But perhaps one of their best properties is that they have a very low equivalent series resistance (ESR) compared to other electrolytic capacitors types like aluminum electrolytic.

The tantalum powder is a dielectric because it easily forms an insulating oxide layer as part of the manufacturing process. The oxide layer thickness is controlled by applying a polarized voltage to the capacitor called the formation voltage which is usually a multiple of the working voltage. The capacitor will also get the same polarity as the formation voltage. It's this oxide layer that is critical to the tantalum capacitor's safe operation.

It's their high capacitance and low ESR that makes tantalum capacitors a good choice in power supplies and for a long time they have been used as a drop in replacement for small aluminum capacitors. But if you do that in a power supply circuit you could have a pretty high failure rate because it turns out that tantalum capacitors are sensitive to voltage transients and ripple current. Tantalum capacitors can absorb small voltage transients within their operating range but if the transients are too large then they can punch through the thin oxide layer on the tantalum powder and damage the capacitor. The capacitor's low ESR can unintentionally amplify the voltage spikes (ringing) and can also cause damaging current spikes. Once a capacitor is damaged it will degrade until it eventually shorts out. If the power source is large enough then the capacitor can actually explode then burn the circuit board. But tantalum capacitors can be used safely as long as you watch their voltage and current ratings. Normally you can get away with using an aluminum capacitor near its maximum rating but for a tantalum capacitor you need to derate its working voltage rating by at least 50%. You also have to make sure that the ripple current won't exceed the capacitor ratings by adding enough capacitors in parallel. The downside to paralleling capacitors is that it also reduces the capacitor ESR, can create a small tank circuit with the inductance in the PCB traces and create unexpected voltage spikes in power supply applications.

Electronics roughly from the late 1980s through mid-1990s used epoxy dipped and surface mount tantalum capacitors quite a bit in their power supplies and for decoupling. Unfortunately these capacitors are now known to have a pretty high failure rate (**Figure 2**). In some cases it's now common advice to test or replace all the tantalum capacitors when repairing 1980/90's electronics. It's hard to know exactly why the failures happened but according to Kemet [1] early tantalum capacitors weren't actually designed for power supply applications and they required a significant amount of external resistance to operate reliably (usually several ohms per volt applies to the capacitor). It's also possible that many designs from that era were using the capacitors near their voltage ratings and that they simply just failed because of many small transient events. For example, it was common to see 6.3 V capacitors used for 5 V power rails.

Tantalum capacitors are being used less often in new designs thanks to MLCC ceramic and better aluminum electrolytic capacitors but if you need a lot of capacitance in a small package tantalums are hard to beat. Modern tantalums have dealt with the problems mentioned here with surge-tested and fused parts. You just have to make sure that that they are used within their limits.

(140366)

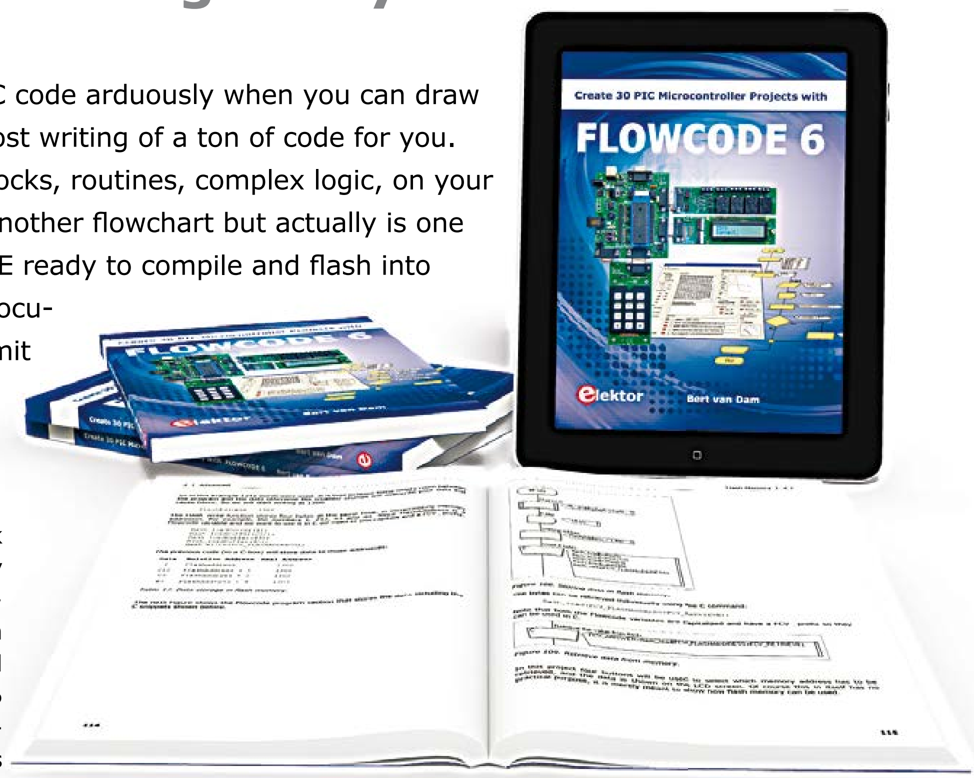
Web Link: [1] http://canada.newark.com/pdfs/techarticles/kemet/Tantalum-in-Power-Supply-Applications.pdf

Create 30 PIC Microcontroller Projects with

FLOWCODE 6

Engineers hate writing—they draw

To drop the question, why write PIC code arduously when you can draw "it" and have a program do the ghost writing of a ton of code for you. Drag and drop symbols, building blocks, routines, complex logic, on your screen, producing what looks like another flowchart but actually is one step beyond already: it's FlowCODE ready to compile and flash into your PIC device. And it gets auto-documented too, so you're good to submit that Science Project, or impress your friends.



Intuitive and fault tolerant as it may look (erasers ready!), the act of drawing may require some examples and encouragement from an instructor. Author Bert van Dam employs a highly structured method of unleashing the power of FLOWCODE 6 for building PIC projects in a totally educational way. Admittedly, to me the chapters did read like drill exercises at times but they have surprises, rewards and niceties in store, providing thrust to keep you going right up to the more challenging stuff.

For example, while the Internet Webserver described in the book may not be a turnkey project to give to your neighbor, it builds perfectly on the chapters that precede it, giving you every confidence not only of your programming skills developed so far, but also of actually knowing how things work down to the last interrupt routine or PIC I/O line.

In terms of hardware to support the projects and do the PIC programming, all that is also down to modular thinking meaning you connect together a number of ready-built E-Blocks for the configuration set out in each chapter. Not all projects require hardware though and I guess if you want to be really creative, you can even build some of the stuff from your own parts.
While editing this book into publication it dawned upon me that if "a picture tells a thousand words", a Flowcode screen tells one Kilobyte. Likewise, an E-Block, one Kilotransistors.

By **Jan Buiting**
(Editor-in-Chief, EIM)

Also Available As E-book

Title: Create 30 PIC Microcontroller Projects with FLOWCODE 6
Author: Bert van Dam
Publisher: Elektor international Media – **ISBN:** 9781907920301 – **Year:** 2014
Purchase: www.elektor.com/flowcode-6-book
Flowcode 6 purchase: www.elektor.com/development/flowcode
E-Blocks purchase: www.elektor.com/development/eblocks

Infinite RGB LED Cube with Flowcode

An Instructables project colorized & enhanced

By **Ben Rowland,**
Matrix Technology Solutions Ltd. (UK)

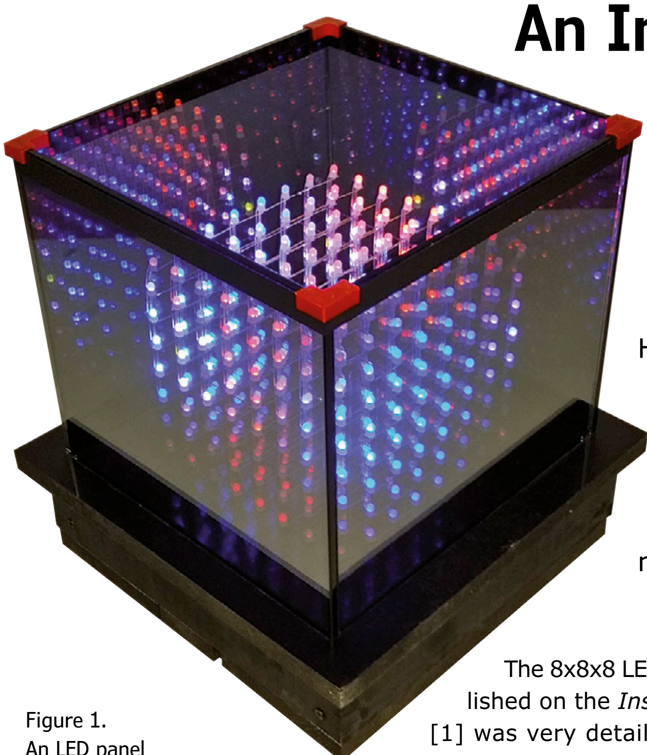


Figure 1.
An LED panel sandwiched between a mirror and a glass pane suggest optical infinity to the LED animations.

Here's how to take a plain single-color 8x8x8 LED cube project to the Next Level. The newly released Flowcode 6 software development suite was instrumental in a lively journey into understanding and achieving the simulation of animations on 'full color' LED cube that's capable of mesmerizing visualizations.

Features

- 512 RGB LEDs in 8x8x8 cube
- Every LED individually addressable
- Optional use of infinity mirror
- Animations can be simulated
- Developed and programmed using Flowcode 6 and ECIO40P16
- Arduino patchable

The 8x8x8 LED cube project published on the *Instructables* website [1] was very detailed and yet looked simple enough for me not just to complete but add substantial enhancements as well.

After planning on creating a larger single color version of the cube I quickly realized that to double the display dimensions to 16x16x16 meant octupling the number of LEDs and interconnections. The thought of processing 2,048 LEDs was too much for me so I scaled down my thinking and instead decided to repeat the 8x8x8 (512-LED) design but using full color RGB LEDs instead of single color LEDs. Even at 8x8x8 RGB LEDs it still tallies up to around 2,200 solder joints so be warned!

From my research I also came across some great infinity mirror projects which are basically LEDs sandwiched between a mirror and a pane of glass with a one-way mirrored surface to create an effect that the LEDs go on forever (**Figure 1**). I decided to apply some of the simple techniques to my cube to try and extend the display beyond the bounds of the actual LEDs.

With 512 RGB LEDs in the design each with their own three internal LEDs a lot of control signals are required to switch the LEDs on and off. Therefore a method is needed to share control signals between LEDs (multiplexing). In the design the LEDs are cathode-grounded with a current limiter resistor on the anode.

Multiplexing tool 1: buffer ICs

The LED cube employs two types of multiplexing to allow all 512 RGB LEDs to be driven independently. The first method uses octal flip flop buffer ICs to allow us to create enough control signals to drive a full 8x8 segment, i.e. 1/8th of the cube. In an 8x8 section there are 64 LEDs which means we need 192 control signals to allow us to drive all the individual color channels. To drive all the signals we need to chain 24 of the buffer ICs together so that the output of the first buffer drives the input of the second buffer, the second drives the third and so on.

Figure 2 shows a simple schematic with two buffers chained together and connected to drive LEDs. The clock signal is driven by the controller and used to shift the output signals from one buffer IC to be passed to the next buffer IC.

I made the buffer driver fairly modular to allow me to scale up or down the size of the cube depending on how many LEDs I wanted. To do this I created a PCB with three buffer ICs on allowing me to control a single row of eight RGB

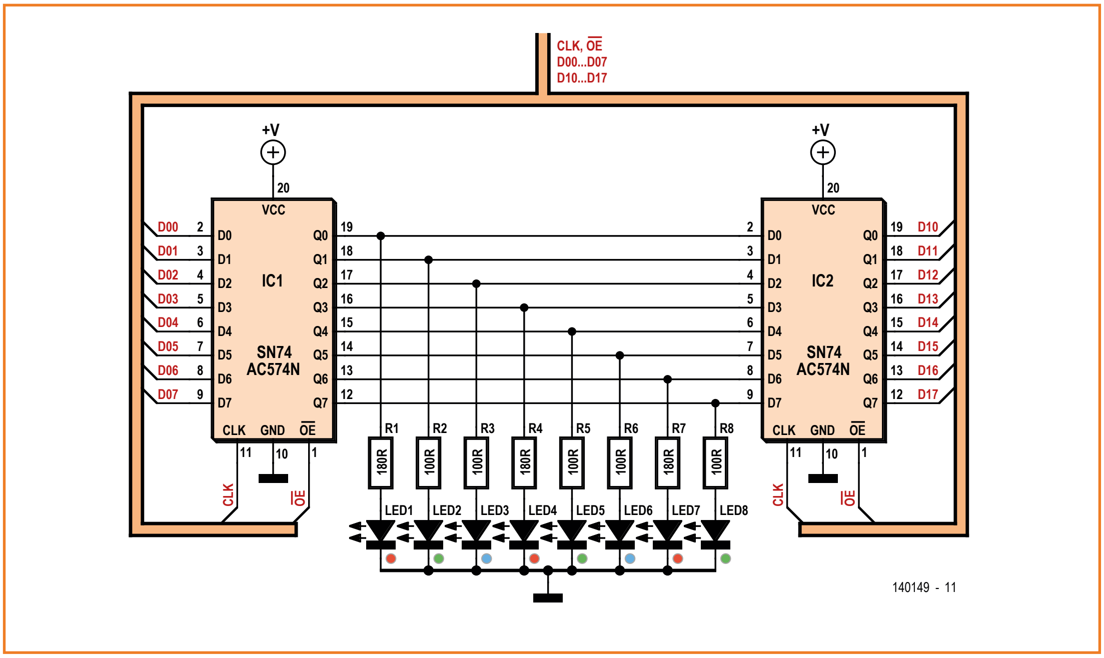


Figure 2.
One method of achieving LED multiplexing is to "insert" the devices between two buffer ICs to which intelligent control is applied.

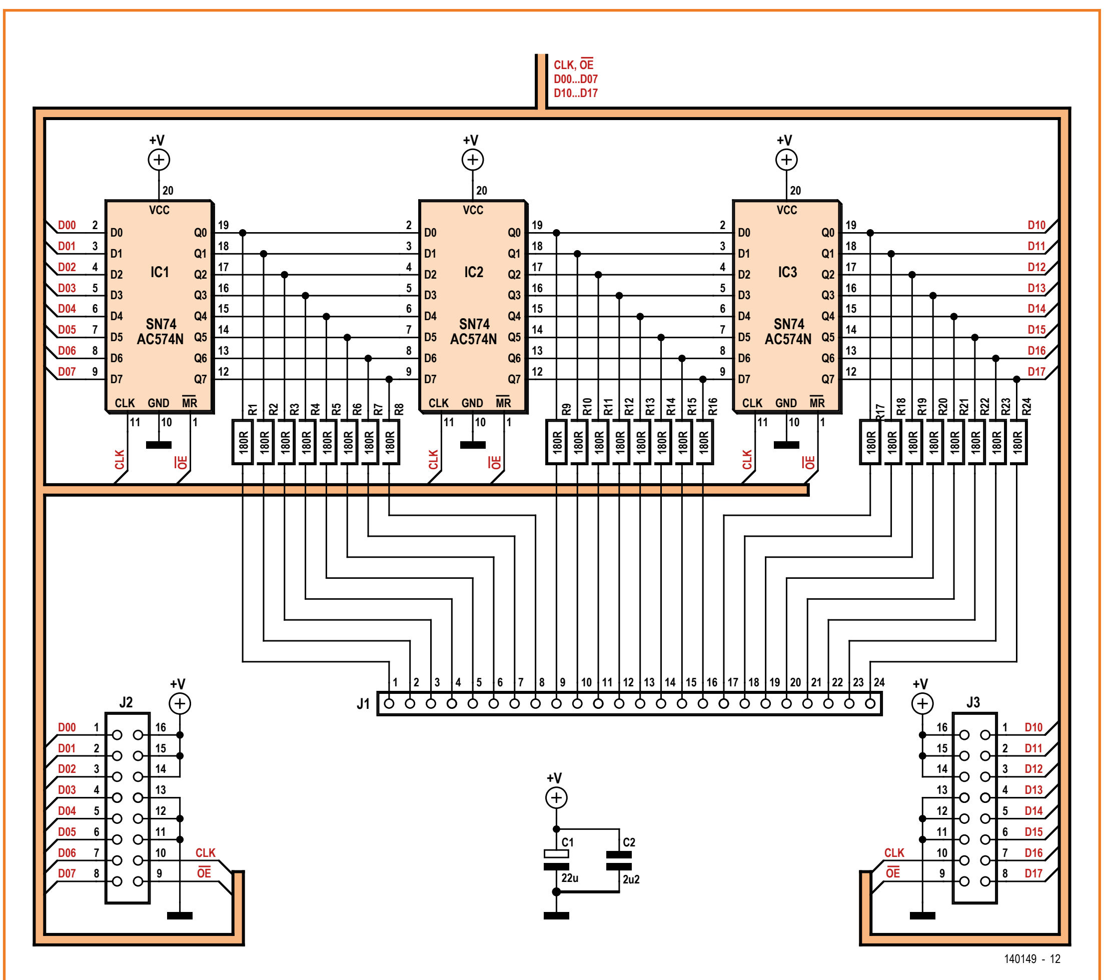


Figure 3.
Schematic of the modular(-ish) buffer board. Eight of these are required for a full implementation of the 8x8x8 RGB Color LED Cube.

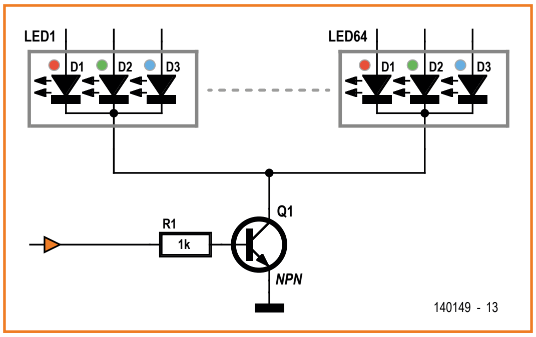


Figure 4.
One of eight NPN transistors used to switch 8x8 sections on and off under software control, completing the overall multiplexing scheme for 512 RGB LEDs.

LEDs. With eight of these driver PCBs chained together it allows me to control the entire 8x8 segment of the display. The smaller 16-way ribbon cable is used to pass the 10 control signals plus power and ground between the PCBs. The wider ribbon cable is used to pass the signals onto the LEDs via the on-board resistors. Each board is fitted with its own capacitors to allow a bit of current to be sourced locally as and when needed by the LEDs.

Multiplexing tool 2: NPN transistors

In **Figure 3** each of the 8x8 LEDs in a single segment is driven via our chain of buffer ICs. To allow us to control the other seven 8x8 segments of the display we need a second method of multiplexing. To do this we take the cathode pins of all the LEDs within the 8x8 segment and connect them all together. We then connect these to ground via an NPN transistor which allows us to enable and disable all the LEDs in the 8x8 segment at once using a single control signal to the transistor's base pin. **Figure 4** shows the approach schematically; **Figure 5** the resulting stack of eight driver boards.

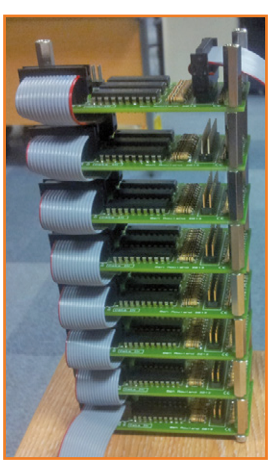


Figure 5.
The stack of driver boards.

By connecting all the LED cathodes together for each horizontal 8x8 section we can then use eight NPN transistors to enable or disable each of the 8x8 sections and therefore control each of the LEDs in each section individually. This is great but obviously this means that we can only light up 1/8th of the display at any one time. By constantly switching which 8x8 section is active at fairly high speed we can build up the entire display so it appears to the human eye that the entire cube is lit and constant.

The Build

So that's pretty much it for the theory, it's time to pull up our socks, make a brew and dive into the world of bending wire and soldering. There are two fairly laborious tasks that are required when building a LED cube, the first being bending LEDs and the second being soldering the bent LEDs together. Each step required a jig to make the process more manageable and to help maintain quality while I slowly lost the will to live. Luckily I managed to do most of this stage during the Christmas vacation so there was enough going on to keep my mind interested while my hands were occupied on the task.

Bending LEDs

The first step for assembly was to take all 512 LEDs and bend them correctly. To do this I drilled a 5mm hole in a spare piece of wood and checked that a LED fit snugly and flush to the wood but could also be inserted and removed easily. Once I had my hole correct I drew some guide lines to help with the bending. As pictured in **Figure 6**, the lines I drew were one vertical running through the center of the hole (A), one horizontal line running through the center of the hole (B) and one horizontal line running through the very edge of the hole (C). A metal ruler came in very handy during the bend stage.

The photos in **Figure 7** show the steps to follow when bending the legs.

1. Insert the LED into the jig ensuring you have got the LED in the correct orientation. The pins should all be sat in line with the horizontal line (B) and the cathode pin should be the second pin from the right.
2. First Bend: bend the cathode pin downwards and the anode pins upwards parallel with the line (A). Ensure they are fairly flat to the board and the bottom of the LED.
3. Second Bend: using the metal ruler bend the cathode pin to the left so it is parallel with line (B).
4. Third Bend: using the metal ruler on the second horizontal line (C), bend the anode pins upwards again. One LED down, 511 left to bend!

Assembling 8x8 sections

Once you have at least 64 bent LEDs you are ready for the next stage which is to assemble one 8x8 section of LEDs. To hold the LEDs in place I created a jig using my trusty 3D printer and a design I uploaded to Thingiverse [2]. If you don't have access to a printer then the Instructables guide [1] describes a different method to hold the LEDs in

place. The LEDs were soldered together by inserting each LED one at a time into the jig before soldering the anodes together on the current column and the cathodes together on the current row as shown in **Figure 8**. Take your time and ensure that every solder joint you make is nice and strong, spending a bit more time here will potentially save a huge amount of time later on when it comes to getting the thing up and running.

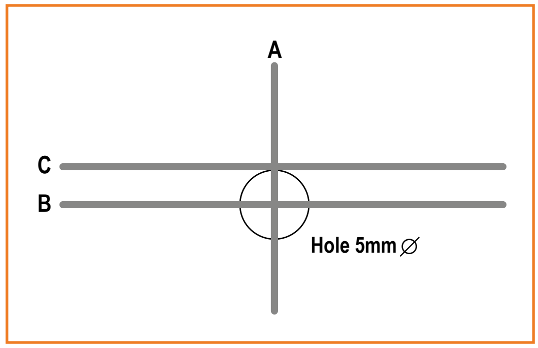


Figure 6.
Guide lines (literally) to help with the LED pin bending.

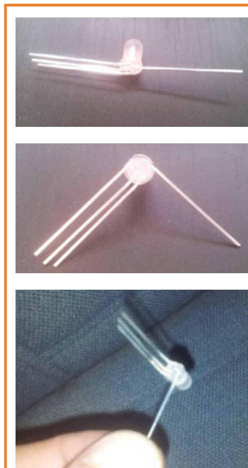


Figure 7.
Method of bending the LED pins for easy assembly into a cube shape.

As each complete panel was finished I made sure to give it a full test while it was still in the jig to ensure that every LED lit correctly and that there were no breaks or shorts in the panel. To do this I connected long lengths of wire to my 5-V power supply. At the end of one of the pieces of wire I soldered the larger of my two resistor values used on the driver board from Port 1 and this allowed me to drive all of the LEDs without any risk of damage. Using my trusty metal ruler pressed against the unconnected common leads I was able to temporarily short all the common pins together to ground meaning I only have to test each anode column once rather than having to test each and every LED individually. To get the panel of LEDs out of the jig I just gently pushed on the top of each top LED in turn and repeated until the entire panel popped loose. Setting the completed panel aside I went back to bending LEDs for the next 8x8 panel.

Combining the 8x8 sections

After all the LED legs were bent and all the LEDs were assembled into panels it was time to start joining the panels together. I laid the first panel flat on a desk and soldered two lengths of straight single core wire to each the cathode wires one at either side of the panel to provide a good level of support and a bit of redundancy just in case one of the solder connections on one of the wires were to fail. To obtain the straight wire I used the technique of holding both ends with a pair of pliers and pulling hard. I had to experiment with a few different types of wire before I found one that would be happy with the current from 64 x 3 LEDs and would also straighten nicely without too much effort on my part. I used a few small match boxes to space the next panel away from the first. Taking my time to ensure everything was as lined up as possible I repeated this process until all of the panels were together and then gave the LEDs another full test to ensure I hadn't missed anything. A virtual 3D view of the cube in the making like in **Figure 9** shows the position of the 8x8 section.

Preparing the base

The cube once fully assembled is fairly strong but all it would take would be a bit of weight applied in a wrong place and all that work would be ruined. I used the same piece of MDF board that I used to create my jigs and drilled 5-mm holes into this at the correct pitch for the legs of

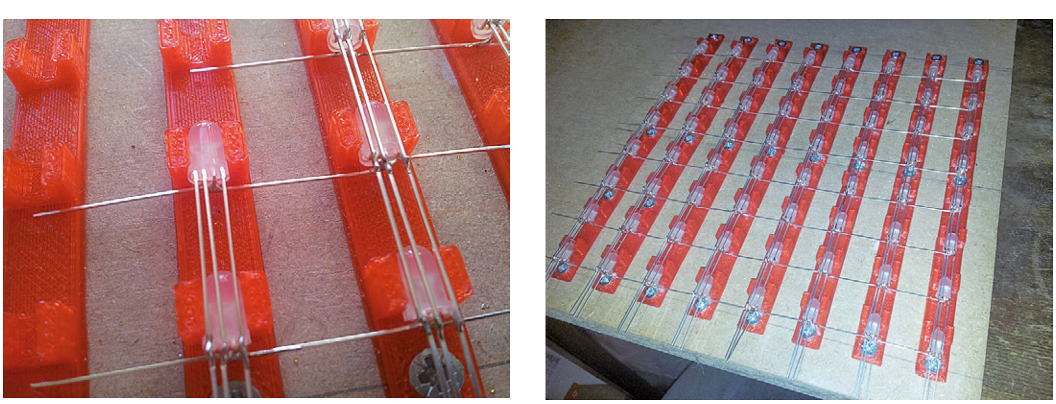


Figure 8.
A jig is used to hold the 8x8 LED arrays together for the purpose of soldering the anodes and cathodes thus creating rows and columns.

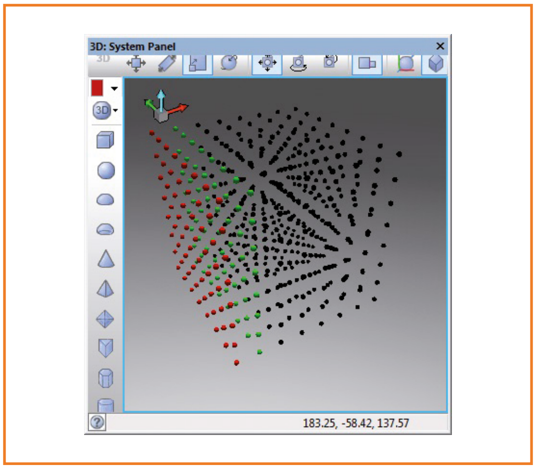


Figure 9.
A 3D rendering showing the whereabouts of an 8x8 section within the cube.

the bottom row of LEDs to go through the MDF. To work out the pitch I took measurements from outer left LED to outer right LED on both sides of the cube and then averaged the spacing, then repeated for front to back. I then drilled a further eight 3mm holes along one edge for the cathode wires to come through to the cube. Finally I routed a 6-mm groove for my glass cover to fit into and spray painted the MDF.

Details on inserting the cube onto the base, adding the common-cathode rails and the project wiring may be found in a supplementary document available for free downloading from the Elektor magazine website [3].

The Brain

I used some pieces of veroboard to allow me to connect an ECIO40P16 'brick' directly to my 5-V power supply and to create the NPN common driver circuitry. A 5-V power supply must be used to allow a fairly large current to flow without generating any additional heat. The Flowcode LED Cube component is flexible so will work with pretty much any microcontroller family apart from 8-bit PICs as these cannot have arrays greater than 256 bytes in size. The ECIO40P16 packs a lot of punch for its size but with suitable code patching a standard AVR based Arduino should also be able to drive the cube if that's what you have available.

Whatever method you choose of packaging or securing the control electronics to the cube assembly, remember to add a hole to allow your power supply to be connected. Some of the popular SMPSUs come in small metal cases so this could also be included inside the box to neaten things up a bit. You may also want to leave the USB cable connected to the ECIO to allow you to a) reprogram without having to re-open the casing and b) for communications via USB should you want to go down this route using the Flowcode USB components.

Enter Flowcode 6

To create the simulation and the code to drive the display I used the Flowcode v6 software and this had everything I needed to allow me to create a masterpiece. I began by creating a sphere on the panel, making it invisible and then cloning the sphere enough times to match the X, Y and Z dimension properties. I then went about creating routines allowing me to get and set the color of each individual sphere. This then allowed me to generate the line and cuboid drawing routines as well as the slightly more complex shift and rotation macros. By using a previous graphical LCD component I was able to rip out the text drawing macro and by finally adding a double buffering system the simulation side of the component was about feature complete.

To add the embedded functionality I created an array of 16-bit variables to store each of the LED color values in the cube. I then created a routine which could be called repeatedly to handle things like clocking out the data to the buffers and switching between the eight common channels. Finally all that was left to do was to add code into the get and set color macros to allow the array to be read and written when not running via the simulation. The final component is now available [3] which can simply be dragged

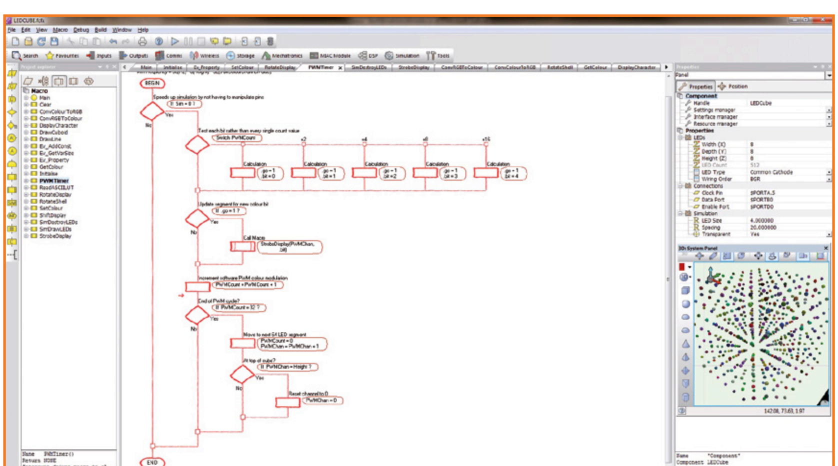


Figure 10.
The program for the RGB LED cube was developed using Flowcode 6 shown in action here. Note the simulation in the right hand pane. The full program and more is contained in archive file 140149-W.zip downloadable at [3].

Web Links

[1] Instructables: www.instructables.com/id/Led-Cube-8x8x8/

[2] Thingiverse: www.thingiverse.com/thing:231031

[3] BOM, CAD, 3D Print, Flowcode 6 Program, Demos: www.elektor.com/140149

[4] Video: https://www.youtube.com/watch?v=odFljHeCNaYt

Hardware Resources

Flowcode 6: www.elektor.com/flowcode-6-for-pic-fc6cp01nepic

ECIO40P16: www.elektor.com/pic-ecio-40-pin-ecio40p

Literature

Create 30 PIC projects with Flowcode 6: www.elektor.com/flowcode-6-book

onto the Flowcode panel. The Flowcode sketching ‘sheet’ is shown in **Figure 10**.

Visualizations

The Flowcode tool has been used to develop a set of test programs to generate a rain effect with lightning, a fireball, two interacting plasma balls, a textual display and a vector based animation. A video is available on YouTube [4] showing the visualizations running on the hardware. The next step is to add a microphone to the internal circuitry and use this to control the movement of the visualizations using techniques such as FFT to chop up the audio signal into frequency bins. Plenty of challenges there—let us know how you get on.

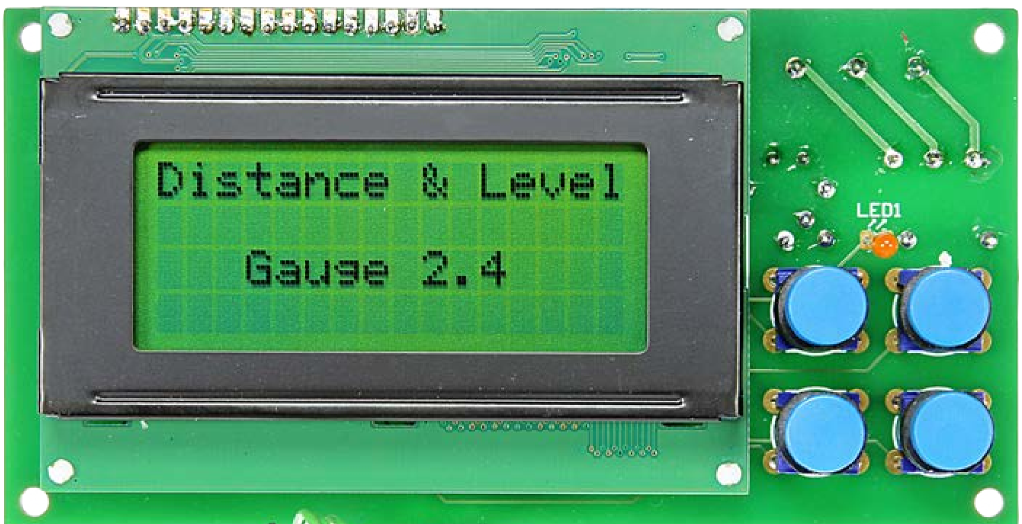
(140149)

Component List

Item Description	Quantity	Source	Part Number
RGB LED, 5mm, diffused lens, common cathode	512	eBay	
5V 5A switch mode DC power supply	1	eBay	
Microcontroller Brain (ECIO40P16 or Arduino)	1	Elektor Store	
SN74AC574N octal flip flop	24	Farnell	165-2006
16-Way IDC ribbon Cable	2 m / 7 ft	Farnell	120-7442
24-Way IDC ribbon cable	4 m / 13 ft	Farnell	120-7439
10µF aluminum electrolytic capacitor	8	Farnell	211-2751
100nF ceramic disk capacitor	8	Farnell	187-1027
100Ω 0.25W resistor (green/blue LEDs)	128	Farnell	933-9043
180Ω 0.25W Resistor (red LEDs)	64	Farnell	933-9230
16-way IDC socket	16	Farnell	221-5235
16-way DIL header	16	Farnell	102-2240
24-way IDC socket	8	Farnell	221-5253
24-way DIL header	8	Farnell	159-3448
TIP122 NPN 5A transistor	8	Farnell	980-4021
Small piece of Veroboard	1	Farnell	120-1473
Tempered glass shelving 6mm / 0.25’’ thick	5	B&Q / Lowe / Home Depot / DIY Store	
Sheet of fiberboard / MDF 20mm/ 0.8” thick	1	B&Q / Lowe / Home Depot / DIY Store	
Spray paint to color MDF	1	B&Q / Lowe / Home Depot / DIY Store	
Reflective car window tint	1	Halfords / Auto Parts	

Distance and Level Gauge

With Alarm Function



By **Jörg Trautmann** (Germany)

Small, widely available and dirt-cheap ultrasonic sensors enable us to design both simple and lavish measuring devices for all kinds of ranging. Take one of these modules, add a display, a couple of pressbuttons and a microcontroller loaded with software, and you now have all the ingredients for a circuit of this kind.

Do you need to know how full your oil tank or water tank is, without requiring vast computational effort or expert knowledge? You fill a large barrel with water from the garden hose and would like to cut off the water flow using a magnetic valve when it has reached the desired level? Or maybe you would just like to measure the distance between two objects? The possibilities using the level and range meter described here are virtually limitless!

Features

- Measures the level status of liquids
- Watchdog function monitors fill level, with relay output and alarm LED
- Programming of min./max. alarm levels infinitely variable
- Stores min./max. calibration values for up to ten tanks or containers in memory
- Also measures distance
- Intuitive menu guidance using LCD display

The extent to which cylindrical or cuboid containers are full can be determined either mechanically with a float and potentiometer, or else capacitively with ultrasonics or a laser device. Both approaches, mechanical and capacitive, require auxiliary components such as floats and sensors. If accuracy of measurement is your prime consideration a laser is definitely the best solution, yet even this method has its pitfalls. For exam-

ple, fog and steam can cause reflections for the laser, falsifying the result.
Since influences of this kind affect ultrasonics hardly at all, I settled on ultrasonic sensors for this cost-effective device. A further advantage of this solution is that it requires no physical contact to be made with the liquid medium. Into the bargain we also obtain a tool for measuring distance.

The frequency of 40 kHz normally used by ultrasonic transducers corresponds, at an ambient temperature of 20 °C, to a wavelength of 8.5 mm. Sound waves in this frequency spectrum do not disperse to any great extent and diverge (in a club-shaped form) with an aperture angle of around 15°. Perfect qualifications for our purpose then, since the side walls of the container will have effectively no influence on the reflected signal.
The ultrasonic sensor used here was already described exhaustively in Elektor a few months ago in the 2014 Project Generator issue [1]. Accordingly we will keep the functional description short and to the point. A trigger pulse causes the ultrasonic module to transmit a burst signal. This signal is reflected by the object to be measured, say the water surface in a rainwater barrel, and received back by the ultrasonic module as an echo signal. Since the sound traverses the distance to the reflecting object twice during the echo period, we can state that:

distance =
0.5 x *velocity of sound* [m/s] x *echo period* [s]

Circuit and components

The schematic in **Figure 1** is based on three components, the well-loved ATmega8 microcontroller, a LCD connected to this plus the ultrasonic module. As well as the HC-SR04 module (**Figure 2**) you can also use the almost identical

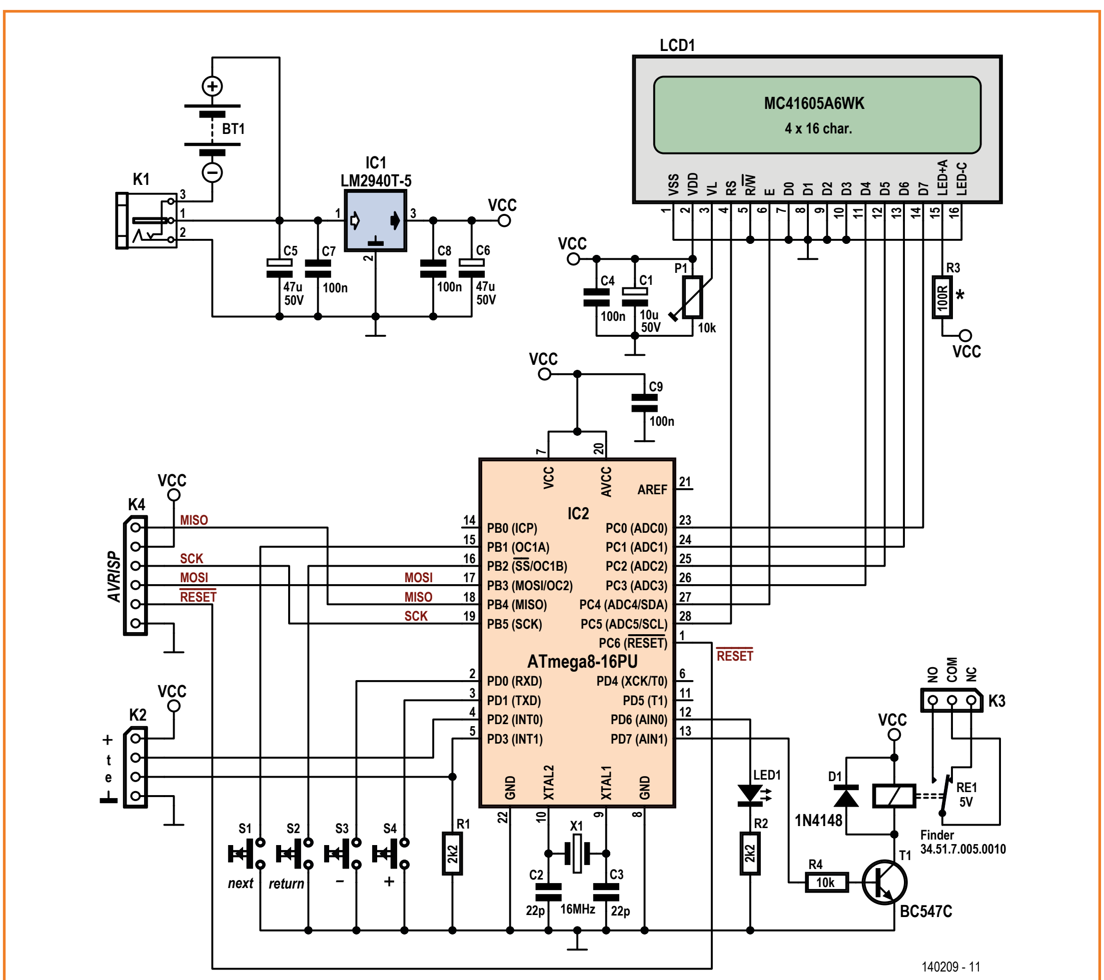


Figure 1.
Simplified circuit of the Distance and Level Gauge.

Figure 2.
The compact ultrasonic module HC-SR04.

US-020. Both modules are in the $5 / £3 price class, making them unbeatably attractive for use in a do-it-yourself measurement device.

Ascertaining the manufacturers of these modules is not an easy task but they are widely available in electronics stores, particularly those that specialize in robotics. When you check out the data sheets, however, you will discover wide variations. One time the detection distance for the two models will be shown as 2 cm–3 m (.8 inch – 10 feet), another time as 3 cm – 7 m (1.2 inch – 23 feet). Further investigation will uncover other divergent details, even for one and the self-same model. Hence the current consumption will vary between 3 mA and 15 mA depending on which data sheet you read. Reliable data can be determined only by doing your own research. Generally suppliers of these modules agree on the following data:

- Operating voltage: 5 V DC
- Temperature range: 0–70 °C
- Diffusion angle: 15°
- Operating frequency: 40 kHz
- Trigger input signal: 10 µs

The minimum distance between transmitter and receiver capsules for the HC-SR04 module is so small that measurements can be made in canisters (the ultrasonic module needs to be discharged when dealing with highly inflammable fluids, owing to risk of explosion!). If you are taking measurements in containers with particularly small aperture diameters you can resort to the SRF02 ([4] ultrasonic module, which although not pin-compatible, is nevertheless equipped with a serial/I²C interface and uses a single capsule for both transmitting and receiving. As the transmit/receive angle is very acute, with this model you need to bear in mind that the minimum distance from the medium being measured amounts to 16 cm.

To simplify the menu guidance a four-line, 16-character LCD display is employed. Potentiometer P1 adjusts the contrast. R3 is the series resistor for back-illumination and should amount to around

R3 = $(5 - V_f) / I_{nom}$ [Ω].

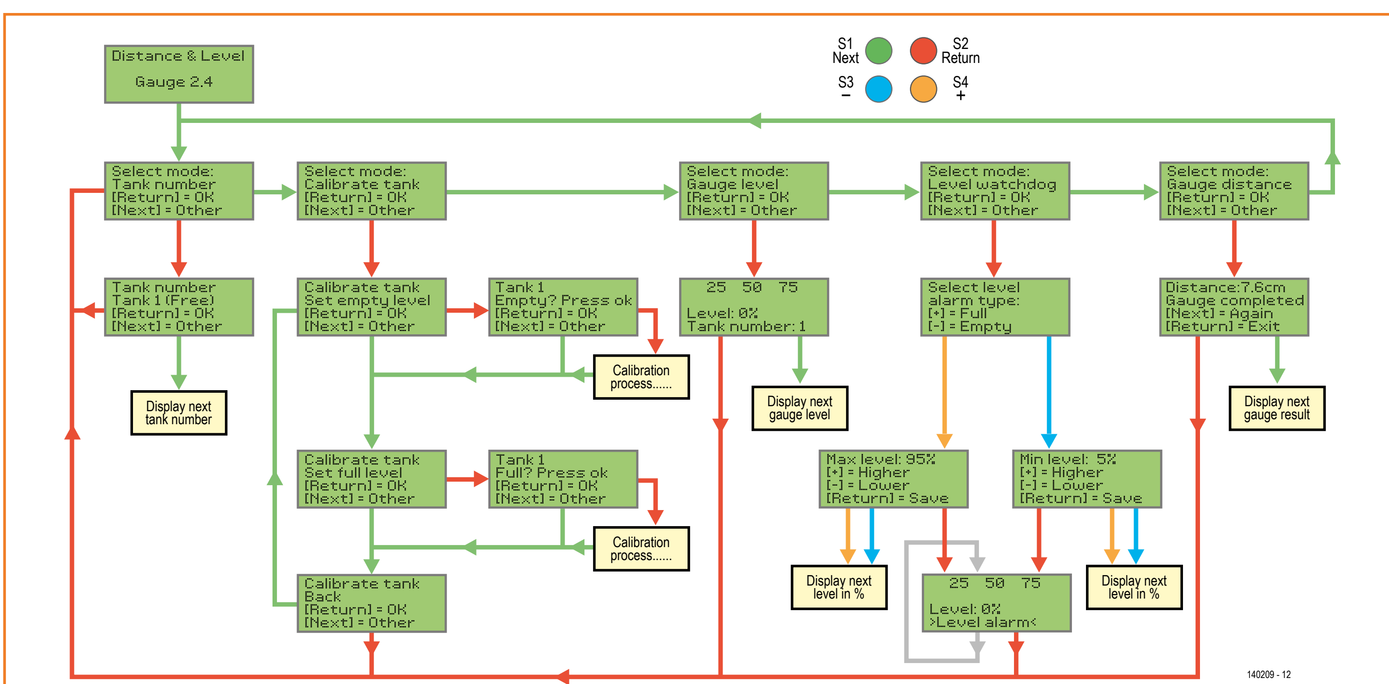


Figure 3.
Procedural diagram for the software.

For this you need to consult the data sheet of the display. With many LCDs the series resistor is already provided internally and in that case you can replace R3 with a wire bridge or strap. To save energy you might also consider a setup involving a switch.

The changeover relay RE1, whose contacts are connected to K3, are controlled by port pin PD7 and driver transistor T1, and the alarm LED (low-current) by port pin PD6. A low-dropout voltage regulator LM2940CT-5 ensures a stable 5 V supply voltage from four 1.5 V batteries. This somewhat exotic voltage regulator delivers up to 1 A and can therefore be loaded much more heavily than the usual 78L05 with its 100 mA. Naturally you can also use a plug-in ('wall wart') AC power supply (<26 VDC) for this project, connected to K1. With the relay energized and the LCD illuminated the circuit draws around 150 mA. To achieve maximum accuracy of measurement, the microcontroller needs an external 16 MHz crystal. A couple of passive components complete the circuitry.

How temperature influences the velocity of sound

The velocity of sound c_0 in air amounts to 331.5 m/s measured at a temperature of T = 0°C. The velocity of sound is not dependent on atmospheric pressure but purely on the air temperature ϑ in degrees Celsius:

$$c_\vartheta = c_0 \cdot \sqrt{1+\alpha \cdot \vartheta}$$

where α is the coefficient of expansion of 1/273.15 = 3.661x10^{-3} 1/°C. From this arises the approximation:

$$c_\vartheta = 20.063 \cdot \sqrt{\vartheta + 273.15} \text{ m/s}$$

We can see that for precision measurement the influence of temperature is by no means insignificant. Here are some examples of sound propagation times at 0 °C and 20 °C.

Distance in cm	Time elapsed in ms at 20°C	Time elapsed in ms at 0°C
2	0.117	0.121
10	0.583	0.603
50	2.915	3.017
100	5.831	6.033
200	11.662	12.066
300	17.492	18.100

The software

The program in the microcontroller must cover the following functions:

- Indication of the level of liquids
- Watchdog monitoring of this level with relay output and LED alert
- Infinitely variable programming of the min/max alarm level
- Storage of min/max calibration values for up to ten containers or tanks
- Distance measurement
- Intuitive menu guidance using LCD readout
- Offset correction for distance measurement
- Manual divisor correction to accommodate low and high temperature extremes

The software follows the procedure shown in **Figure 3**. In view of the limited amount of memory space in the ATmega8, readout messages are given in English only. Operation makes use of the four pressbuttons *Return*, *Next*, *Plus* and *Minus*. Because of the complexity of the program coding, which can be downloaded along with the hex file and the PCB layout at [2], it has been subdivided into meaningful procedures and provided with corresponding commentaries, so that even beginners should find their way through it rapidly.

The actual measurement process follows. To bring the ultrasonic module connected to K2 into transmitting a signal, a pulse 10 µs long with a falling edge is sent via port pin PD2 to the trigger input. As a result, about 250 µs afterwards, the ultrasonic module transmits a burst signal with a frequency of 40 kHz and duration of 200 µs. The echo output, which is connected to the microcontroller via port pin PD3, switches now to High level and the ultrasonic module stands by to receive the reflected signal. If this occurs, the echo output flips back to Low. Timer1 is ticking away throughout this process, so that after the timer is halted, the distance can be calculated. After 50 ms if no echo signal arrives, measurement for the current cycle is cancelled. To achieve the best possible results the mensuration cycle involves 16 separate measurements. The program code for this can be examined in the procedure `Gauge_distance()`.

Now a word of advice for people doing their own 'burning' and taking advantage of the programming interface for the controller at K4. Because

the program is extremely wide-ranging, the 8 K-capacity flash memory is 100 % full. For this reason the demo version of BASCOM-AVR cannot be used for compiling, since it allows only 4 K (i.e. 50%) of memory usage. While burning you can rapidly blunder into a trap: in this circuit the ATmega8 operates with an external 16 MHz crystal, making it necessary to set two fuses. `CKOPT` should be activated to assure reliable operation of the oscillator. This method increases current consumption somewhat but you do obtain stable oscillation in return. It's essential to set `SUT_CKSEL` to `Ext. Crystal/Resonator High Freq.: Start-up time: 1K CK + 64 ms`! The list of options provides a wide selection of settings. If you make a mistake here, under some circumstances you can 'brick' your ATmega8 once programmed, as it may well be unable to find its feet again afterwards. Better then to double check precisely which settings you have selected. A screendump of AVR Studio showing the correct fuse settings is included in the download package.

Construction and commissioning

Construction is extremely simple thanks to the very manageable number of components. You can buy the double-sided printed circuit board (**Figure 4**) from Elektor or etch it yourself at home. The relevant files are in the download package [2]. Some handiwork and skill are necessary to create cutouts in the case for the LCD readout, the ultrasonic capsules, the pressbuttons and the switch. How you arrange the ultrasonic capsules is a matter of personal taste. Installing the ultra-

Component List

Resistors
R1,R2 = 2.2kΩ
R3 = 0Ω (wire link)
R4 = 10kΩ
P1 = 10kΩ trimpot

Capacitors
C1 = 10µF 50V, 2mm pitch
C2,C3 = 22pF 50V, 0.1" pitch, C0G/NP0
C4,C7,C8,C9 = 100F 50V, 20%
C5,C6 = 47µF 50V, 0.1" pitch

Semiconductors
D1 = 1N4148
LED1 = low current, red, 3mm
T1 = BC547C
IC1 = LM2940T-4.0
IC2 = ATmega8-16PU, programmed, Elektor Store # 140209-41

Miscellaneous
K1 = DC-power socket, 1.95mm pin (Farnell 1217037)
K2 = 4-pin pinheader
K3 = 3-way PCB terminal block, 0.2" pitch
K4 = 6-pin (2x3) pinheader with jumper
LCD1 = LCD module, 4 lines 16 characters
RE1 = 5V, 1 c/o (Finder 34.51.7.005.0010; Farnell 1169338)
S1–S4 = pushbutton, Multimec RA3FTL6 (Farnell 1132885)
4 caps for Multimec 1D03 (Farnell 1132887)
X1 = 16MHz quartz crystal, CL =18 pF
HC-SR04, US-020, or SRF02U ultrasonic module

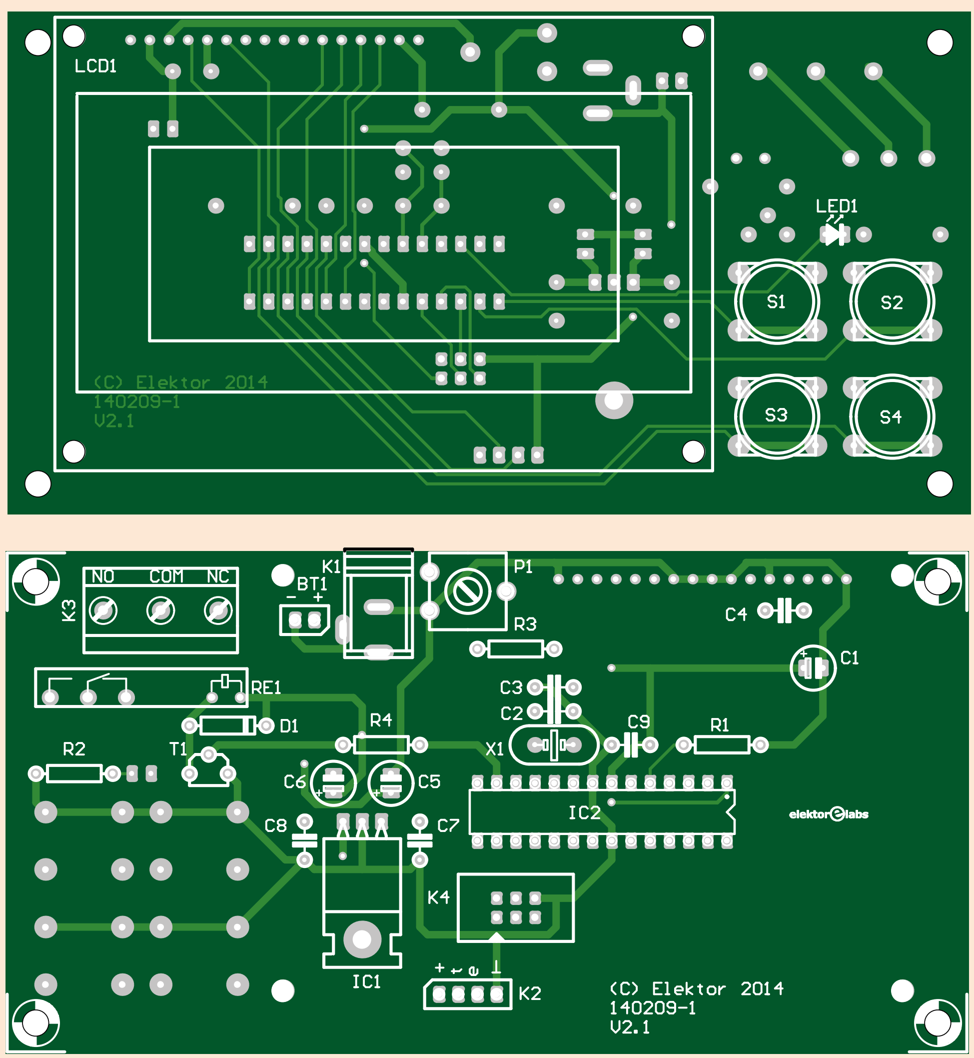


Figure 4.
Upper and lower sides of the PCB.

sonic modules separately on a moderately long screened cable is also conceivable, in case reading needs to take place in another room.

Once the programmed ATmega8 has been inserted and the ultrasonic modules connected, you can switch on the device for the first time. At very least the background lighting for the LCD should illuminate. If this is not the case, disconnect the power supply and double-check the component placing! During the start phase *Distance & Level Gauge 2.4* (**Figure 5**) should be visible on the display for two seconds.

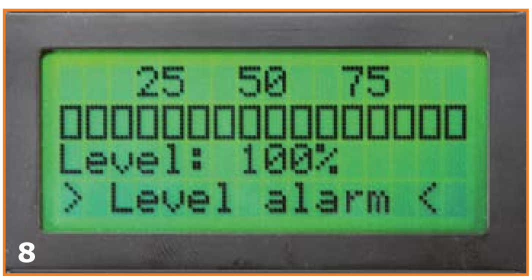
Now you can perform the basic settings. To do this you switch off the device, then turn it on again while holding down the *Return* button. Having released the button, you will be asked the width of the container being measured. At this point you can press the *Plus* and *Minus* buttons in order to enter an offset in 5 mm increments for taking into account the distance from the top of the ultrasonic module to the leading edge of the container. This gives you a simple way, for example, of measuring the distance from one wall to the other later on. The best method for this is to take a 2 m reference measurement with a two-meter wooden rule (or measure three feet using a carpenter's yardstick). The value is confirmed using *Return*.

5

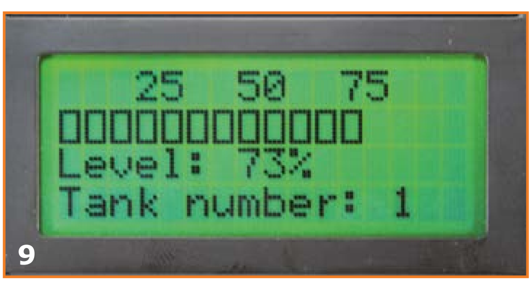
Following this you are asked for the divisor value used for correcting the value measured. This is necessary since the velocity of sound is temperature-dependent. In the temperature range from −20°C to +40°C the velocity of sound varies from 312.85 m/s up to 349.32 m/s. The deviation is virtually linear, so that for a temperature variation of 10°C you can expect an inaccuracy of about 2%. **Panel 1** contains some formulae and a table of values for the elapsed times to

6

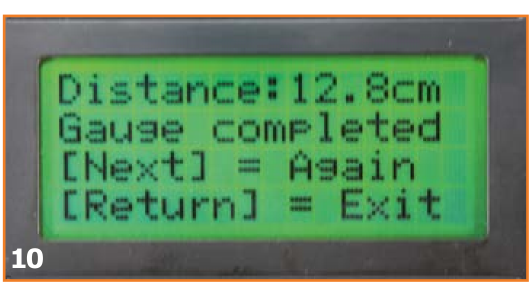
7



8



9



10

be expected. Incidentally, atmospheric pressure has no influence on the velocity of sound.

Anyone prepared to accept up to 5% inaccuracy in this connection need do nothing at all. On the other hand, if you wish to take measurements in temperatures below 15 °C or above 25 °C and with a higher level of accuracy, a division factor must be introduced. The higher the temperature, the greater the divisor must be. The values for the width of the container and the divisor must also be retained after a battery change and can be entered afresh at any time.

The *Select mode:* menu should now be visible. Pressing the *Return* button will take you to the tank selection menu. The opportunity is provided to carry out calibration functions for up to ten different tanks or other containers. The only condition made is that the container shape must be as close as possible to a cuboid or cylindrical form, otherwise linear measurement is impossible. Having selected your tank number you land back at the main menu. After pressing the *Next* button the menu option *Calibrate tank* appears. Hitting *Return* now goes to the calibration menu. Having selected empty-level calibration, you are asked if you are ready to perform the calibration. You don't necessarily need to empty the vessel in question for this purpose; instead you can simply measure outside the tank the height from the filler neck down to the ground.

Once the empty level has been calibrated, you reach the menu for full-level calibration (**Figure 6**). The default value is set as a minimum of 3 cm, which you can accept by pressing *Next*. If, however, you wish to perform a calibration for the full-level, put the device in position above the filled container and press *Return.*

After returning to the main menu you can perform the first level measurement. To do this, select menu option *Gauge level* and confirm with *Return*. Measurement begins now and the result is displayed. *Next* carries out a fresh measurement, *Return* leads back to the main menu.

To test out level monitoring you should select the *Level Watchdog* menu. In the menu that follows (**Figure 7**) the alarm type can be selected: *Full* will activate the alarm when the permitted fill quantity is exceeded and *Empty* when it falls short. Having chosen your alarm type, you can alter the threshold value in the *Level Alarm* menu (**Figure 8**). For the default values we have defined the maximum level as 95 % and the minimum level as 5 %, but these can be varied using the *Plus* and *Minus* buttons. All the calibration values made are stored in the EEPROM and are retained there even after a change of battery.

The first measurement follows next and the current level is displayed (**Figure 9**). Measurements are now taken every five seconds. If these fall below the minimum value or exceed the maximum value set, the display shows > *Level alarm* <, the red LED illuminates and the relay operates, for example to switch on a pump. If the level status returns into the permitted range after 5 s, the alarm function is cancelled. The 5 s measurement cycle avoids the need for a hysteresis function to eliminate relay contact bounce.

To exit from this mode back to the main menu, press the *Return* button (for up to 5 s). You can now select the *Gauge distance* menu. Each press on the *Next* button initiates a distance measurement (**Figure 10**). As before, you can exit from this mode and go back to the main menu by pressing *Return*.

(140209)

Web Links

[1] www.elektor-magazine.com/130546

[2] www.elektor-magazine.com/140209

[3] HC-SR04: www.cytron.com.my; SRF02: www.dfrobot.com/wiki/index.php/SRF02_Ultrasonic_sensor_%28SKU:SEN0005%29

ORDERING INFORMATION

To order, contact customer service for your region:

USA / CANADA
Elektor US
111 Founders Plaza, Suite 300
East Hartford, CT 06108
USA
Phone: 860.289.0800
E-mail: service@elektor.com

Customer service hours:
Monday-Friday 8:30 AM-4:30 PM EST.

UK / ROW
Elektor International Media
78 York Street
London W1H 1DP
United Kingdom
Phone: (+44) (0)20 7692 8344
E-mail: service@elektor.com

Customer service hours:
Monday-Thursday 9:00 AM-5:00 PM CET.

PLEASE NOTE: *While we strive to provide the best possible information in this issue, pricing and availability are subject to change without notice. To find out about current pricing and stock, please call or email customer service for your region.*

COMPONENTS

Components for projects appearing in Elektor are usually available from certain advertisers in the magazine. If difficulties in obtaining components are suspected, a source will
normally be identified in the article. Please note, however, that the source(s) given is (are) not exclusive.

TERMS OF BUSINESS

Shipping Note:
All orders will be shipped from Europe. Please allow 2–4 weeks for delivery.

Returns
Damaged or miss-shipped goods may be returned for replacement or refund. All returns must have an RA #. Call or email customer service to receive an RA# before returning the merchandise and be sure to put the RA# on the outside of the package. Please save shipping materials for possible carrier inspection. Requests for RA# must be received 30 days from invoice.

Patents
Patent protection may exist with respect to circuits, devices, components, and items described in our books, magazines, online publications and presentations. Elektor accepts no responsibility or liability for failing to identify such patent or other protection.

Copyright
All drawings, photographs, articles, printed circuit boards, programmed integrated circuits, discs, and software carriers published in our books and magazines (other than in third-party advertisements) are copyrighted and may not be reproduced (or stored in any sort of retrieval system) without written permission from Elektor. Notwithstanding, printed circuit boards may be produced for private and educational use without prior permission.

Limitation of liability
Elektor shall not be liable in contract, tort, or otherwise, for any loss or damage suffered by the purchaser whatsoever or howsoever arising out of, or in connection with, the supply of goods or services by Elektor other than to supply goods as described or, at the option of Elektor, to refund the purchaser any money paid with respect to the goods.

MEMBERSHIPS

Membership renewals and change of address should be sent to the Elektor Membership Department for your region:

USA / CANADA
Elektor USA
P.O. Box 462228
Escondido, CA 92046
Phone: 800-269-6301
E-mail: elektor@pcspublink.com

UK / ROW
Elektor International Media
78 York Street
London W1H 1DP
United Kingdom
Phone: (+44) (0)20 7692 8344
E-mail: service@elektor.com

Do you want to become an Elektor GREEN or GOLD Member or does your current Membership expire soon?

Go to www.elektor.com/member.

Improved Current Transformer

By J.T. van Es (Netherlands)

This report is a response to the two publications by Martin Ossmann in the April and May 2014 editions about the dimensioning and use of current transformers. Martin aims to reduce the lower cut-off frequency, determined by *L*/*R*, by looking at the core material, among other things. While the method is not incorrect, similar results can be achieved much easier as shown in the report below.

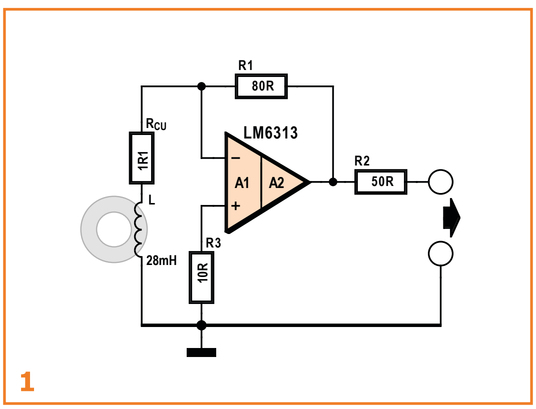


Figure 1.
Basic circuit.

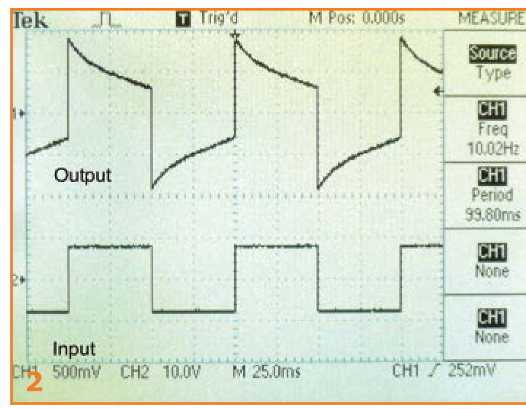


Figure 2.
Output signal (10 Hz) without positive feedback.

Figure 1 shows the signal path of a current meter I built in 2005 and which was appropriate for its purpose. It ended up in a closet after that, but was retrieved again for this occasion. The current transformer is a pick-up coil from a 'Tubantor' fuel-injection system from 1983. This contains a ferrite core with 80 turns. The -3 dB bandwidth of this assembly is 7 Hz to 5.5 MHz. Using it with a 10-Hz square wave is therefore not such a success, as **Figure 2** shows. R3 has been added to correct the offset of the amplifier.

By giving the amplifier a little bit of positive feedback, using R4 and R5 (see **Figure 3**), the input resistance of the amplifier becomes negative (adjustable with R5) and this effectively compensates for R_{Cu}. When the circuit resistance

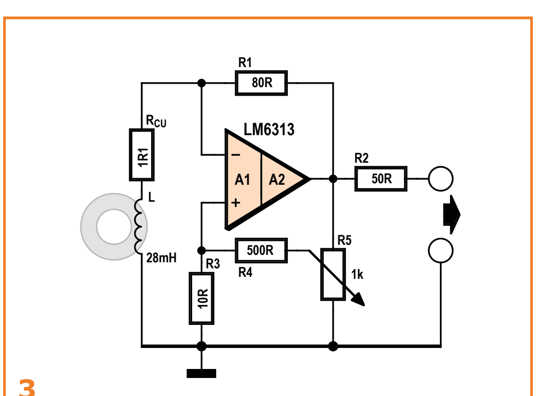


Figure 3.
Positive feedback via R4/R5.

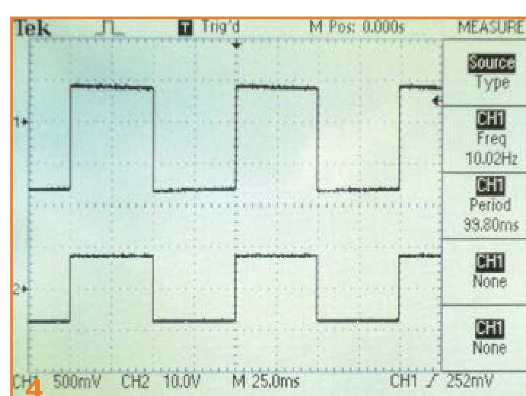


Figure 4.
Output signal (10 Hz) with positive feedback.

reduces, L/R increases. **Figure 4** shows this effect. It even works well down to 1 Hz, if the positive feedback is increased a little more, see **Figure 5**. However, the DC stability leaves much to be desired. **Figure 6** shows a solution for this, using a control circuit comprising an additional opamp, R7 and C1. The opamp needs to be a type with FET inputs so that it becomes possible to work with large values for R7 and C1. C1 needs to be a film capacitor in order to keep the leakage current low.

Because the secondary winding is almost perfectly short-circuited using this technique, there can be no voltage drop across it any more—and not across the primary either, apart from its own copper resistance. The flux in the core, which is already low in a current transformer, also approaches the absolute minimum. The value for L can be much lower in this case, which is beneficial for its HF behavior. The measurement of DC signals is, however, not possible, this requires a Hall effect sensor. The annoying feature of such a device is that it requires a gap in the core, which results in a much reduced mu and therefore value of L.

In the past (1985, I think) we designed a beam monitor using a Vitrovac ring core with an inside diameter big enough to pass a beer bottle through. The bandwidth we obtained then was 23 mHz to 20 MHz!

(140266)

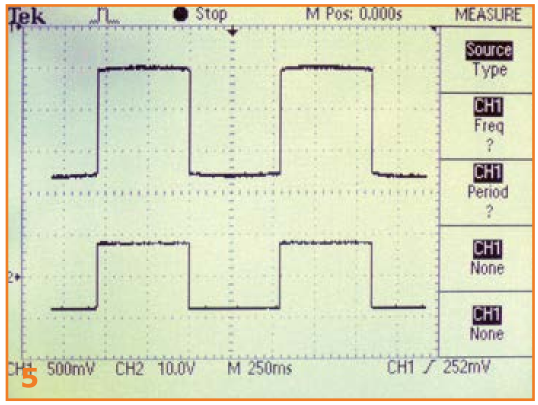


Figure 5.
Output signal (1 Hz) with more positive feedback.

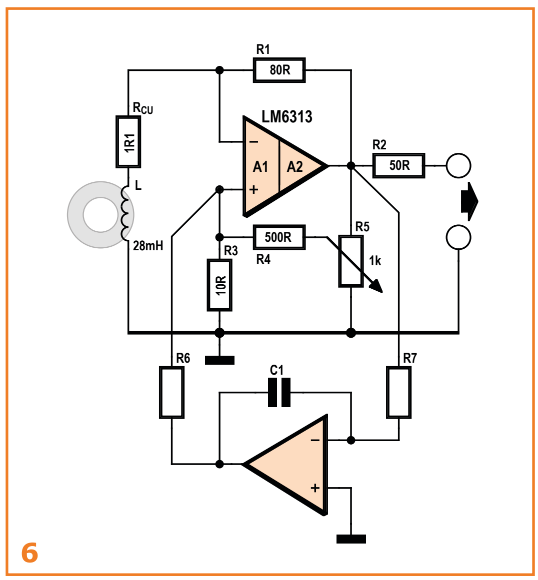
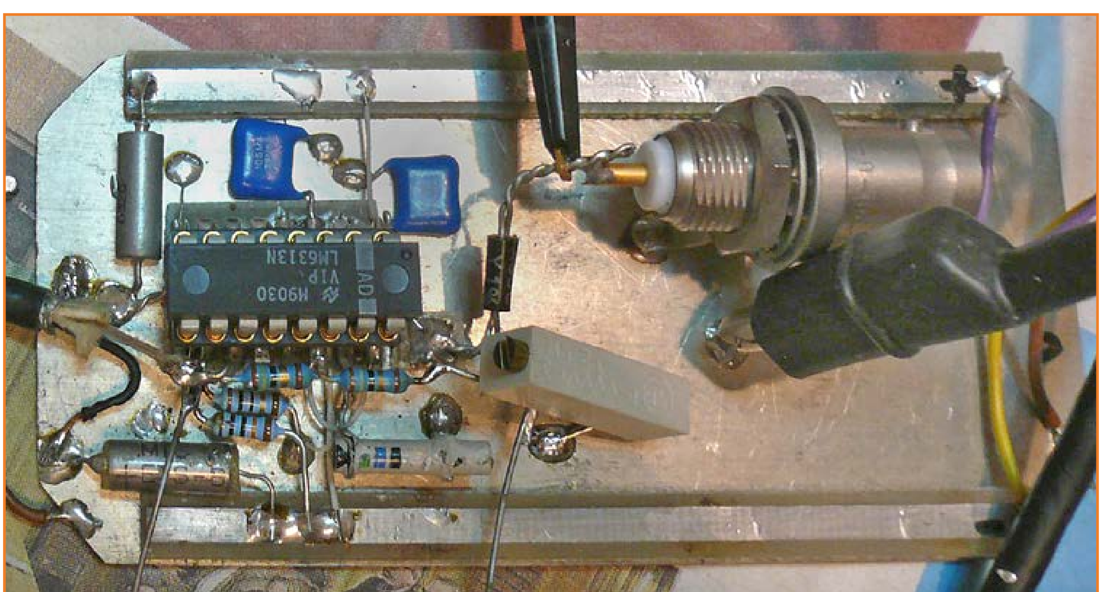


Figure 6.
Improved DC-stability with an additional opamp and R7/C1.



The author's amplifier setup.

Safe High Voltage Power Transistor Measuring with Oscilloscopes

Saelig's CLIPPER CLP1500V15A1 is a high technology oscilloscope adapter that allows small voltages to be measured in the presence of very high voltages, such as those present on switching MOSFETs or IGBTs (Insulated Gate Bipolar Transistors). Making accurate measurements in the presence of very high voltages has previously been extremely challenging, but using the CLIPPER CLP1500V15A1, the V_{dsON} or R_{dsON} of a switching transistor can now be easily seen in high resolution (e.g. 100 mV/div) on an ordinary digital oscilloscope. Determining the switching speeds, Safe Operating Area (SOA), and temperature effects of MOSFETs and IGBTs is now greatly simplified with the CLIPPER CLP1500V15A1.

The CLIPPER CLP1500V15A1 represents a leap forward in transistor switch testing, essential both in the design verification phase and production process monitoring. In typical use, the CLIPPER is connected between Drain and Source terminals of a Power MOSFET. The output signal provides a 1-to-1 picture of what is occurring across the MOSFET, without the disturbing influence of the high drain voltage. In this way, the oscilloscope resolution can be reduced to 0.2V per division, giving excellent visibility of the on-time switching characteristics.

The CLIPPER can directly measure the voltage between Drain and Source (V_{ds}) of a power transistor (MOSFET or IGBT). The output voltage measurement is limited to a safe range between 0 and 2 V (Lo Range) or 0 to 14 V (Hi Range). This allows the oscilloscope to be set to the appropriate full-scale range, with a sensitivity of 200 mV or 2 V per division. The high resolution Vds information obtained, can be used for indirect measurements such as:

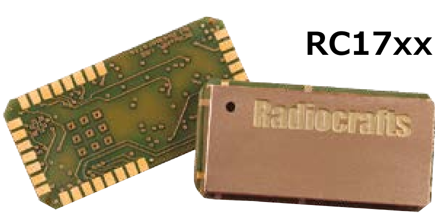
- Measuring R_{dsON} during operation (since switching current is usually known).
- Observing changes to the R_{dsON} during operation, indicating thermal resistance changes (detecting insufficient heatsinking or cooling).
- Junction temperature — directly visualized based on R_{dsON}.
- Thermal stability — directly visible as the load is dynamically varied

For thermal process monitoring, if the R_{dsON} is not stable (i.e. a non-stable SOA), then a reliability issue exists. This could be a catastrophic thermal runaway issue, resulting in shutdown, or the presence of a higher than expected junction temperature (T_j), reducing the transistor lifetime.

In the verification phase of a product, design issues causing SOA problems can include: incorrect load regulation, incorrect transistor selection, gate drive optimization problems, IGBT turn-on issues, or poor thermal design. These issues and others caused by variations in production processes (e.g. soldering quality, PCB thermal and heatsink conductivity, or cooling problems) can all benefit from using the CLIPPER.

The CLIPPER is ideally suited to: production screening, 100% product evaluation, electrical and thermal design verification and qualification, repair and maintenance. No calibration or external power is required for the CLIPPER CLP1500V15A1, which comes supplied in an attractive aluminum transport case, together with an assortment of SMA and BNC adapters.

www.saelig.com (140386-VI)



RC17xxHP Narrowband RF Modules

Radiocrafts AS provides a new radio module series for the industrial radio market. The RC17xxHP series provides narrowband radio performance combined with high output power for long range industrial applications. The new module series covers all ISM bands below 1 GHz, including 169, 433 and 868 MHz, as well as some national bands used for application specific radio communication (444, 458, 465 MHz, etc.). The embedded RC232 radio protocol provide point-to-point and point-to-multipoint communication over an encrypted data link, and is very easy to use.The narrowband modules operate in bandwidths down to 12.5 kHz, with temperature stabilized crystal oscillators to ensure reliable operation over the full industrial temperature range, −40 to +85 degrees Celsius. The RF output power is configurable up to 500 mW. The combination of superior sensitivity of the narrowband radio, and high RF output power gives a very long communication range. The selectivity and blocking properties of these radios makes them ideal for use in critical applications and environments with radio noise from industrial equipment.

The new module series is intended for applications like crane controls, industrial sensors and other high reliability radio links. The very good selectivity and blocking properties provide possibility to work reliably in environments with strong RF interferers.

The modules come in a very compact 12.8 x 25.4 mm form factor, in surface mount package, delivered on Tape&Reel. Samples are available now.

www.radiocrafts.com (140482-II)

RN4020 Bluetooth® Smart Module

Microchip Technology's first Bluetooth® 4.1 Low Energy module type RN4020 builds on the Company's deep Bluetooth Classic experience and carries both worldwide regulatory certifications and is Bluetooth Special Interest Group (SIG) certified. The integrated Bluetooth Low Energy (BTLE) stack and onboard support for the common SIG low-energy profiles speeds time to market while ensuring Bluetooth compatibility, eliminating expensive certification costs and reducing development risks. The module is also pre-loaded with the Microchip Low-energy Data Profile (MLDP), which enables designers to easily stream any type of data across the BTLE link.

Because the RN4020 is a stack-on-board module, it can connect to any microcontroller with a UART interface, including hundreds of PIC® MCUs, or it can operate standalone without an MCU for basic data collection and communication, such as a beacon or sensor. This standalone operation is facilitated by Microchip's unique no-compile scripting, which allows module configuration via a simple ASCII command interface—no tools or compiling are required.

The designers of cost-sensitive embedded applications are looking for turnkey solutions that make it easy to add the low power consumption and simplicity of Bluetooth LE connectivity, which enables several years of operation from a single battery and has a large installed base of compliant smartphones, tablets and computers. Example markets that need these low-power wireless command-and-control solutions include home automation and appliances; medical and wearable devices; toys, tags, fobs and remote controls; pulse and proximity sensor-based systems; and even industrial applications.

Microchip's RN4020 Bluetooth LE Smart module includes all of the hardware, software and certifications that designers need to easily add this low-energy connectivity to any design, while easing End Product Listing (EPL) via QDID Bluetooth compatibility testing. All of the programmable profiles are stored and selectable on the module, including Microchip's flexible MLDP and the common Bluetooth SIG low-energy profiles. In addition to common public profiles, private services can be created via the ASCII command interface. The RN4020 also provides a built-in PCB antenna with 7 dBm transmit power and a receive sensitivity of –92.5 dBm, enabling operation over 100 meters in a compact form factor of only 11.5 x 19.5 x 2.5 millimeters.

www.microchip.com/get/07BX (140482-I)

Stereolithography-based 3D printer

RS Components (RS), the trading brand of Electrocomponents plc has added the small-format production-quality micro-SLA ProJet® 1200 from 3D Systems to its range of rapid-prototyping 3D printers.

The ProJet 1200 low-cost professional-grade printer is the first product in the RS 3D printer range to use stereolithography (SLA) technology, which delivers a smoother finish and builds layers that are less visible compared to alternative low-cost 3D printer technologies such as Fused Filament Fabrication (FFF).

In addition to building prototype concepts, the ProJet 1200 is ideal for small, precise and detail-rich casting patterns. Factory calibrated for reliable and accurate operation, the micro-SLA ProJet 1200 integrates enhanced LED DLP (Digital Light Processing) laser technology to print 30-micron (0.03 mm) layers at a resolution of 56 microns (effective 585 dpi), which results in extremely fine feature detail and smoothness and makes it suitable for building production components such as electronic connectors. The printer also boasts a diverse range of uses outside of engineering markets, such as jewelry making or in dental applications.

Designed for bench-top use, the low-maintenance printer measures 230 x 230 x 356mm and offers a build volume of 43 x 27 x 150mm and a vertical build speed of 14 mm per hour. The printer integrates a UV curing station and an all-in-one material cartridge with built-in print window. The printer uses 3D Systems' VisiJet® FTX Green material, which is a durable and rigid material that was created especially for the ProJet 1200 for high-quality plastic prototyping and casting patterns. The material burns out cleanly in the UV curing process to leave ash-free castings.The machine is simple to set up with an installation process on a Windows-based PC much like a standard 2D printer with easy connection to LAN networks. The software also offers built-in STL CAD file format verification.

The 3D Systems ProJet 1200 and VisiJet FTX Green cartridges are available now from RS stock in EMEA and Asia Pacific regions. Prices are £2860 for the 3D printer and £258 for a pack of 10 cartridges.

www.rs-components.com (140482-V)

One Station for all Soldering & Rework Tasks

Weller's The new WXR 3 Rework Station is designed to handle all tasks related to soldering, desoldering, and the use of hot air in your production process. This multifunctional station is notable for its high cost efficiency. It saves significant acquisition costs compared to buying individual units. As a member of the WX family, the station offers full WX tool compatibility, tried-and-tested benchtop functions and all essential traceability features. The WXR 3 is a 3-channel station with an enormous total power output of 600 watts—more than any other station on the market. The intelligent power management system automatically controls the power output of the individual tools and thus saves a tremendous amount of energy. The available power is distributed to the connected tools as required—a total of 600 watts. High-performance tools can be operated at the same time. 3 channels, 200 Watts each.

The multifunctional USB port allows quick and mobile configuration. The WRX 3 has all the standard user-friendly features you have come to expect from Weller products: backlit graphic display, multilingual menu navigation and operating status display. A large selection of connectable tools, such as soldering irons, preheating plates, desoldering tools or solder baths, leaves nothing to be desired. Two integrated pumps make the system independent of a separate compressed air supply. A high-performance pump for vacuum and hot air, and a separate vacuum pump for pick-up tools.

www.weller.de (140482-III)

New GPS Disciplined OCXO

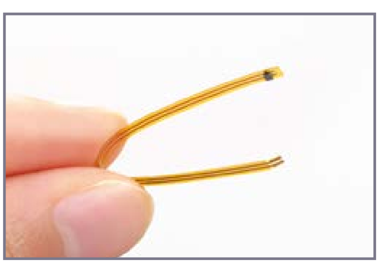
IQD's newest addition to its range of advanced oscillator modules is its IQCM-110 series of GPS disciplined OCXOs. Unveiled at *electronica 2014*, the new design incorporates an internal GPS receiver with a 1-pps output and is housed in a 14-pin 60-mm square package. When coupled to an external aerial via the incorporated SMA connector, in the event of the loss of the GPS signal the highly specified 10-MHz OCXO will switch-in with a holdover capability of 1.5 µs for a 24-hour period thereby maintaining lock until restoration of the reference signal.

The standard operating temperature range of the module is –20 to 75 degrees C but other temperature ranges and holdover specifications can be considered upon request. The required power supply is 5 V with the output being standard HCMOS. Current consumption is 2 A maximum during warm-up with this reducing to 1 A once the steady-state condition is reached.

The design incorporates an internal adaptive algorithm which enables the module to 'learn' the parameters of the GPS signal after a period of 2 days of lock so that the holdover function can start in the event of signal failure. An internal alarm is built-in to indicate lock failure and subsequent restoration of signal. In addition, the unit incorporates a serial connection for more detailed interrogation of the device's performance.

Intended for a range of applications including land based telecommunications systems as well as marine based navigation systems the IQCM-110 forms part of a range of highly specified modules and Oven Controlled Crystal Oscillators available from IQD.

www.iqdfrequencyproducts.com (140482-IV)



Flexible Film Temperature Sensors for Wearable Products

Murata today announced that is about to commence mass production of a range of surface-mounted NTC temperature sensors that are packaged on a flexible printed circuit (FPC) film. With an FPC thickness of approximately 100 µm, they can be easy routed inside complex designs and tight spaces. Owing to their low heat capacity, the sensor's thermal responsiveness is excellent. Measuring 50.00 x 3.17 x 0.55 mm the sensors are ideal for sensing the housing temperature of smartphones and tablets. They are also very suitable for use in wearable products for sensing body surface temperature.

The FTNT55XH103FA1A050 sensor can measure temperatures in the range of –40 to +125 degrees C and has an accuracy, at 25 degrees C of ±0.4 degrees C. Resistance at 25 degrees C is 10 kΩ ±1%.

As smartphones and tablets have become more compact with even greater sophisticated functions the need to monitor the thermal design in space-constrained designs is more important. Also, the Internet-of-Things (IoT) trend of wearable products is gaining popularity and measuring the user's body temperature is just one of the features of products such as fitness bands and sports performance monitors. The new low-profile film temperature sensor is ideal for use in these and many other applications.

www.murata.com (140482-VII)

Two Philips GM2308 Audio Signal Generators (1950, 1964)

Just beat it!

By **Jan Buiting,** Editor-in-Chief

In the dark pre digital age it was hard to make a laboratory-grade audio signal generator that was affordable, stable, accurate, low in harmonics, and lightweight all at the same time. In the early 1950s the scientists at Philips Physics Labs, Holland could be relied on to come up with slightly quaint but technically charming solutions.

Just for the fun of it I run a small lab at home equipped for the most part with vintage Philips (not: *Phillips*) test and measurement equipment. I can only assume that I have developed the same habit as electronics workers 40-60 years ago in switching on the power of all relevant equipment to allow it to heat up while I head downstairs for a coffee. As you can see on the cover of my book [1] almost all test equipment I run is tube based and it's reassuring to encounter the typical smell and the soft glow of the instrument lights when I re-enter the workplace. My instruments do not hum noticeably as is often assumed, and no thump is heard when the central AC comes on. All is properly fused and isolated.

Recently however I was greeted by a foul smell from a small AF generator type GM2315 which had blown "something" inside. I had not been too happy with the instrument recently and it was on my list for inspection and repair.

I recalled I had a GM2308 AF signal generator in the attic and decided to use it as a functional drop in for the 2315. Sadly I had forgotten the 2308 is almost twice the size and weight of the 2315 so getting it installed was accompanied by sounds neither sinewave nor suitable for printing.

GM2308 Features

- 0 – 16,000 Hz frequency range
- MiniWatt tube complement
- 0-25 V output voltage
- Symmetrical or asymmetrical output voltage
- Accurate internal output attenuator (max. 10^4)
- Frequency calibration with electron beam indicator
- Externally accessible output amplifier / attenuator
- Low hum, noise and distortion
- Low effect of AC line fluctuation
- Suitable for use in tropical regions
- Weight only 29 lbs
- Power consumption only 50 watts

Purpose & history

Both the GM2315 and the GM2308 are audio-frequency sinewave-only signal generators for use in laboratories and repair shops. They differ vastly in terms of accuracy, range and price. While the 2315 goes a tad over 20,000 Hz the 2308 is strictly limited to 16,000 Hz. Not a strange boundary though, considering that in the early 1950s there was no such thing as 22,000 Hz as the upper frequency limit for audio gear, let alone HiFi as a concept or design target. When the GM2308 was designed around 1950, 16 kHz was deemed enough for all practical purposes as far as electroacoustics research was concerned. The GM2308 also has a much more accurate and powerful output and attenuator section than the 2315, it can supply up to 25 volts out with decent accuracy in four ranges, while on the 2315 the output level is guesswork really, as is the frequency setting.

The GM2308 was originally designed around 1950 and with its two huge dials was a prominent member of Philips' GM series of lab equipment easily identified by the stern black front panels, large knurled knobs, grey steel cases and leather carrying handles (that snapped invariably). The instrument also appears in some *PTR* photos taken inside the renowned Eindhoven/Waalwijk Physics Labs, specifically in the acoustic research rooms.

Mint vs. battered

When a friend noticed the mint grey & silver GM2308 on my bench as it was being powered up slowly on a variac, he said he also had "a thing like that with two big dials but all black". And sure it was a GM2308 too, a much earlier version than mine judging from the black front panel. The thing was dug out of the garage and given a coarse wipe to clear the worst grime. Apparently the older instrument had been dropped or squeezed in a tight place as it had dents, scratches and crooked control spindles. I also gave it a slow start and noticed no problems except a virtually dead magic eye.

Just beat it!

Wanting to improve on the stability and accuracy of the familiar RC and Wien oscillators of the time Philips exploited the principle of making two high frequencies interfere on purpose, resulting in a beat frequency that's the difference between the two. Suppose you mix 99 kHz with 100 kHz you get 1.00 kHz. Just as on that old AM radio in the evening when the foreign station just 1 kHz off "your" frequency causes a whistle. But why go through so much trouble building two HF oscillators, a mixer, low-pass filtering and more stuff when you can generate 1,000 Hz off the bat with R's, C's and one tube?

The HF mixer approach is worth the extra mile because LC oscillators have intrinsically lower distortion than their RC counterparts. But hey an L at say 100 Hz is a monster and very difficult to pull as little as 10% around that—the effort would require an impossible tuning capacitor. So we build two LC oscillators, one variable between 85 kHz and 100 kHz and another, between 100 and 101 kHz. And tune the two for *beat frequency*.

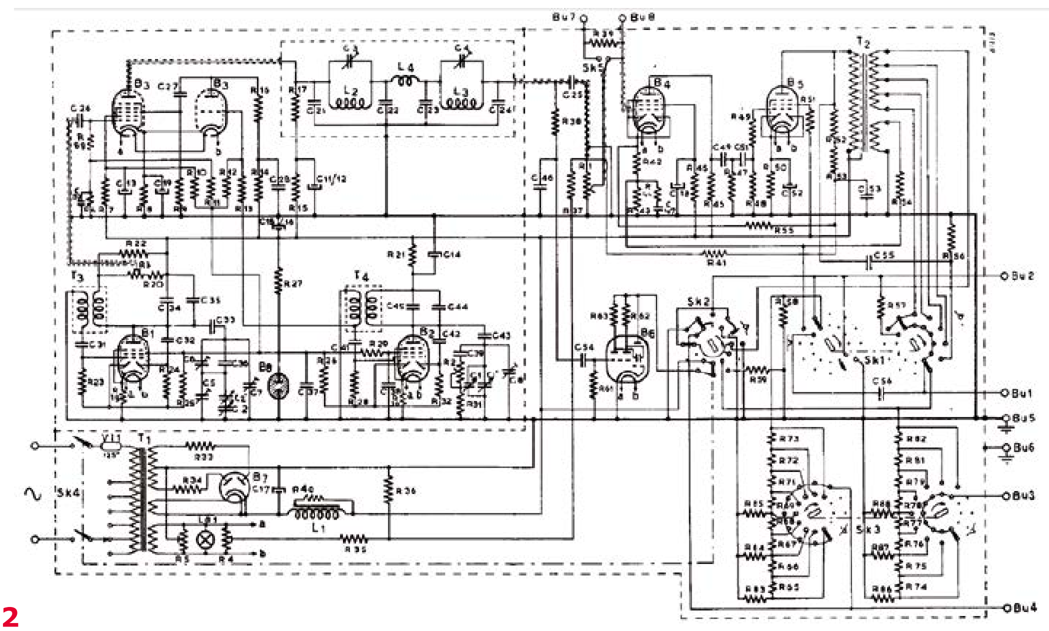
1

Mathematically, mixing two frequencies f_1 and f_2 by applying them to an active device like a tube yields the following products in the anode current:

f_1, $2f_1$, $3f_1$ …
f_2, $2f_2$, $3f_2$, …
f_1+f_2, f_1-f_2, f_1+2f_2, f_1-2f_2, … etc.

In the case of f_1 = 85-100 kHz and f_2 = 100-101 kHz you can see the ease of suppressing the unwanted high-frequency components $2f_x$, $3f_x$, ... with a simple low-pass, the difference with the target 0-16,000 audio frequency range being enormous.

If 100 kHz can be called high frequency, the Philips company had considerable experience with HF design thanks to their world renowned tube radio products and coil winding technology for mass production. Not surprisingly in the GM2308 Philips' the distinctive beehive trimmers appear in the service-friendly position on top of huge coils, the LC assembly meticulously screened by a metal can (**Figure 1**).



2

Not the Evil Eye

The schematic of the GM2308 pictured in **Figure 2** has few surprises. Tubes B1 and B2 are the 85-100 kHz and 100-101 kHz oscillators respectively, and C2 and C1 the tuning capacitors associated with the large frequency dials on the front panel of te instrument. Note R2 which allows very light tuning of the 100-101 kHz oscillator, of the order of hertz. Mixing occurs in B3, which is followed by the expected low-pass filter. At the output is B6 the EM34 magic eye (“electron beam indicator”), which being direct coupled to the LPF shows the frequency difference between B1 and B2. To calibrate the instrument before every use, the big dials are turned to their home position, next R2 is turned with *fingerspitzengefühl* until the EM34 starts to blink, eventually the image will be still, or move ever so slowly “like breathing”, see **Figure 3**. In this way, f_1 and f_2 can be matched within less than 1 hertz. The zero calibration will drift with time however and has to be redone every 30 minutes says Philips. It’s not that bad—even if the EM34 shows butterfly activity the error is minimal at about 1 Hz. BU7 and BU8 allow external signals to employ the final amplifier B4-B5. Normally however, the AF signal from the LPF is applied to B4. The two-tube amplifier is a party pooper, its distortion probably exceeding that of the carefully wrought heterodyne signal, see **Table 1**. By default the maximum output power from the attenuator is 625 mW, you can see the dramatic effect on distortion if 1 W is selected.

The extensive output attenuator with its two-deck switch Sk3 covers a range of 40 dB (10,000). Sk2 allows the output to be switch between balanced and unbalanced.

The 6 positions of Sk1 allow you to select output impedances of 5 Ω, 250 Ω, 600 Ω and 1,000 Ω, all carefully matched on the secondary of transformer T2. In position I-90V-ASYM. the voltage from the primary winding is fed straight to the output, this is a mean 90-volts into 100 kΩ min.

EM34 magic-eye tubes have a poor reputation in terms of availability and life expectancy; hence B6 only receives plate voltage in the FREQ. (zero calibration) position of Sk2. Did you know LED-based replacements are sold for the EM34, EM4, EM1, EM35? [2]

Never change a winning team

Look carefully at the photo in **Figure 4** of the two GM2308 chassis side by side and you can see the minimal changes made to the GM2308 over the course of about 10 years. Left is the old GM2308, right the later one. It has the reliable ‘mustard’ capacitors in critical places, indicating that it was produced around 1964, so the GM2308 probably lasted for over 15 years from concept to replacement (by transistors). Compared to that the iPhone’s life span approaches zero and one day may go negative. Note the ring

ESTD 2004

Retronics is a monthly section covering vintage electronics including legendary Elektor designs. Contributions, suggestions and requests are welcome; please telegraph editor@elektor.com

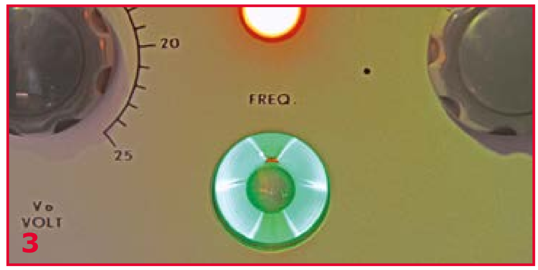
on the capacitors used in the older GM2308 (left chassis view), these indicate the 'earthy' side of the component, or the outside of the dielectric. Also note that the resistors in the right GM2308 are larger and probably have higher wattage to prevent change of value due to moisture absorption, surface cracking or heating. There are more small differences between the two GM2308s. In the newer one I saw one Pope and one Tungsram tube while the old one has an original Philips MiniWatt tube complement only.
The older generator has the isolated jack socket on the back that allows the final tube, an EL84, to be used as an amplifier by driving it from an external source. When a 6.3-mm audio jack is inserted the link between the generator output and the input grid of B4 is broken. Apparently this feature was dropped by early 1960. Little surprise because the GM2308 has no facility for frequency sweeping or bursting. The older generator is, ermm, old but not from the oldest production batch because that apparently had an EBL21 dualdiode-pentode tube as the final. Equipment versions /01 had the EL84.

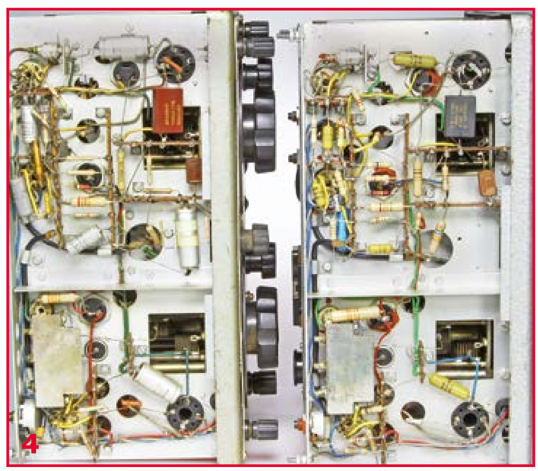
Operation

In mild words, that takes some getting used to. The zero calibration with the EM34 butterfly image is fun and intuitive, not so with the actual frequency setting let alone the output level. To know the output frequency you have to add the frequency dial readouts, like 14,400 + 750 equals, ermm. In practice you turn the left dial to a round kHz value and the right to vary within 1,000 Hz. For the AF output signal level, more elementary mathematics is required, juggling exponents like 3 x 10^{-x} on the stepped output. In practice I guess you just listen to the audio equipment under test when the sinewave hits and only then bother to turn the attenuator controls to get sensible results and not blow up things, although tubes are forgiving.
In my shack, the newer GM2308 was found to meet its original specifications for distortion with ease, while a minimal tweak on a pot was required to calibrate the frequency range.

Table 1. Distortion

Frequency	Max. distortion at P_{out}			Note
	400 mW	625 mW	1 W	
30 – 200 Hz	2%	3%	4%	distortion and hum
200 – 16,000 Hz	0.75%	1%	2%	distortion





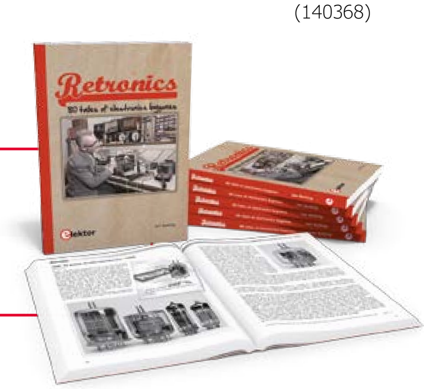
These results are probably due to the mustard caps, the older GM2308 is slightly off the mark in some respects—nothing smokes, it just needs recapping.
Being slightly older than the newer GM2308, I can still hear that 16,000 Hz but not much beyond that. Yet when driven by sinewaves from my GM2308 all my amps, tube and transistor, do really well for frequency response. *No see no hear no speak,* but what's out there beyond 16 kHz?

(140368)

Reference and Web Link

[1] Retronics, 80 Tales of Electronic Bygones, Elektor International Media, ISBN 978-1-907920-18-9

[2] http://www.pcvana-z.nl/ledogen.html



Hexadoku The Original Elektorized Sudoku

With all that shopping, inviting and cooking on the 2DO list for the Christmas period you are well advised to start early on this tough little puzzle here. Find the solution in the gray boxes, submit it to us by email, and you automatically enter the prize draw for one of five Elektor book vouchers.

The Hexadoku puzzle employs numbers in the hexadecimal range 0 through F. In the diagram composed of 16 × 16 boxes, enter numbers such that **all** hexadecimal numbers 0 through F (that's 0-9 and A-F) occur once only in each row, once in each column and in each of the 4×4 boxes (marked by the thicker black lines). A number of clues are given in the puzzle and these determine the start situation.

Correct entries received enter a prize draw. All you need to do is send us **the numbers in the gray boxes**.

Solve Hexadoku and win!

Correct solutions received from the entire Elektor readership automatically enter a prize draw for five Elektor Book Vouchers worth **$70.00 (£40.00 / €50.00)** each, which should encourage all Elektor readers to participate.

Participate!

Before January 1, 2015,
supply your name, street address and the solution (the numbers in the gray boxes) by email to:
hexadoku@elektor.com

Prize winners

The solution of the October 2014 Hexadoku is: **C03EF**.
The €50 / £40 / $70 book vouchers have been awarded to: Philippe Monnard (Switzerland), Ivan Corneillet (USA), Siegfried Kepper (Germany), Steve Hasko (UK), and Jean-Paul Lagaisse (Belgium).
Congratulations everyone!

7	2						5	A						4	9
				6			9	D			5				
5	B	6			3					9			C	D	0
	0		9	4	C					F	1	5		E	
	1	7	A			6	C	E	5			B	4	3	
B	9		0			A			2			6		F	E
	3		8	5		9			6		B	C		1	
				B	D					A	C				
				A	5					1	2				
	6		E	C		1			9		D	8		5	
A	7		1			3			E			9		C	2
	D	F	C			4	0	8	A			E	3	B	
	8		B	D	7					C	0	3		A	
3	A	0			1					2			B	7	F
				F			6	7			3				
6	F						3	9						2	C

0	B	6	7	5	1	D	8	A	3	4	E	9	2	C	F
2	9	8	1	3	A	B	C	5	F	7	0	D	6	E	4
A	E	C	3	4	6	7	F	2	8	9	D	0	1	5	B
D	4	F	5	2	E	0	9	6	B	1	C	3	7	8	A
7	5	B	2	8	9	E	1	C	D	6	4	A	0	F	3
9	8	D	A	6	5	C	0	3	E	F	B	1	4	2	7
C	3	1	6	D	F	4	7	0	2	5	A	B	E	9	8
E	F	0	4	A	B	2	3	9	1	8	7	5	C	D	6
F	0	2	B	7	D	1	A	8	9	E	6	C	3	4	5
1	D	3	9	0	2	F	4	B	7	C	5	6	8	A	E
8	6	4	C	E	3	5	B	D	0	A	F	7	9	1	2
5	7	A	E	9	C	8	6	1	4	2	3	F	B	0	D
6	C	9	F	1	7	3	2	E	A	D	8	4	5	B	0
3	1	E	D	B	4	6	5	F	C	0	2	8	A	7	9
4	2	5	0	C	8	A	D	7	6	B	9	E	F	3	1
B	A	7	8	F	0	9	E	4	5	3	1	2	D	6	C

Appearances Count

By Gerard Fonte (USA)

It's said that you can't judge a book by its cover. But, of course, we do that all the time. We often take a book off the shelf because the cover has caught our eye. It's true that we don't know if it's a good book to read. But the cover sure looks interesting. It's the same in business. If you want to get noticed, you wear your best in order to stand out from the crowd.

The Paper Trail

Documents travel far and wide in any organization. So, when you write your next report, take a little extra time to make it sizzle. First start with a good font. Twelve point, Times Roman is a good choice. This is a standard size and is easy to read on the page. Times Roman is a "serif" font. That means that there are small lines at the ends of the characters. For example, the lower-case "L" will have a short horizontal base with serif fonts. Non-serif fonts (or "sans serif") print the "L" as a plain straight line. These extra small lines add a flourish that tend to be perceived as more elegant or formal than sans serif fonts. Most books are printed with a serif font. Elektor uses a sans serif font. Obviously, no one will say that this is a great paper because it uses a nice font. Rather it will be a subtle influence that the reader will not be aware of. (If it doesn't matter, why are there so many fonts?) It's like combing your hair and polishing your shoes before an interview. You should never use any type of non-traditional or "cutesy" font under any circumstance (that I can think of).

The content must be clear and well written. Write so you can't be misunderstood and target your audience using words that they are familiar with. Don't try to be clever and impress the reader with your vocabulary. Sesquipedalian tergiversation is confusing and usually makes you appear pretentious and arrogant rather than clever and intelligent.

Start the paper with a short summary (called an "overview" or "abstract") with the conclusions or the major points you will discuss in the paper. This accomplishes two important things. The first is that it primes the reader to expect certain topics. Saving the critical items for last is usually a bad idea. The reader can get distracted, or lose track, or simply lack the time or effort to finish it. Obviously, if your paper isn't finished, it won't have much impact. The second reason for placing a summary at the start is because many higher-level managers will only read the summary. They won't plow through the whole paper to find the significant aspects. Their "time is too important." So, providing them with a simple and quick way to get the vital information that they need is something that they will appreciate and remember. Being appreciated and remembered should always be goals of anything you write.

Looking Good

The overall appearance of the paper is important. Make it look like a professional publication. Break up the text with topic headings (like "The Paper Trail" above). I usually make it **bold** and two points larger than the text. People find large blocks of unbroken text daunting. Smaller pieces are easier to digest.

Adding pictures is always a good idea. But put the pictures where they are first mentioned. Don't just add them at the end like an afterthought. Having them conveniently placed is courteous and makes your document easier to read and understand. Usually the pictures can be about a quarter-page or smaller in size. Of course, it depends on the content. If there is text in the picture it must be big enough to read easily (no more than two points smaller than the body text). This may require re-writing that text in a bigger font so that it is still readable when it is reduced. (Sometimes you can add a text-box with larger text over the existing text.) And don't forget to use captions for the figures. Again, many people will just scan your report, so make it easy for them to get the essentials.

Now add a cover-page. The cover usually should only have the title, date, your name and the company name. The title should be a large font and your name and date should be smaller but not less than 16-points. I add a plane square box as a frame about one inch from the outside boarders of the page.

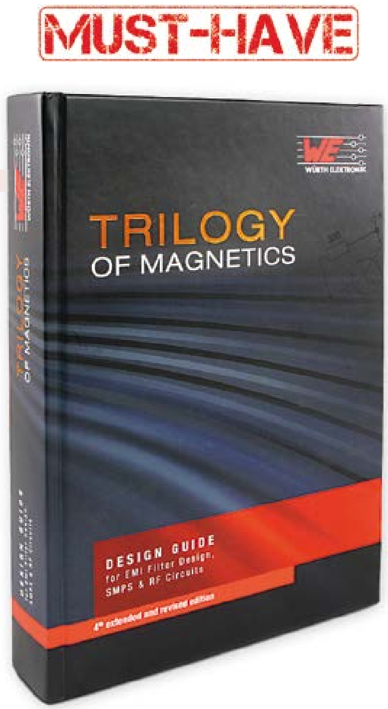
In a Bind

If this is going to be a hard-copy paper it's a huge plus if you can bind it in some way. Most businesses have some means for binding documents. (Getting it printed and bound at a local office-supply/copy-center may be well worth your expense.) I use plastic combs. They're easy and inexpensive. This also assumes that there are a small number of copies required. If you need more than a dozen or so copies, binding it may appear wasteful to some people.

If this is an electronic paper, then convert it to PDF. This appears much more professional that a plain word-processing document. However, be sure to carefully review the conversion. I have seen "conversions" that have many significant differences from the original.

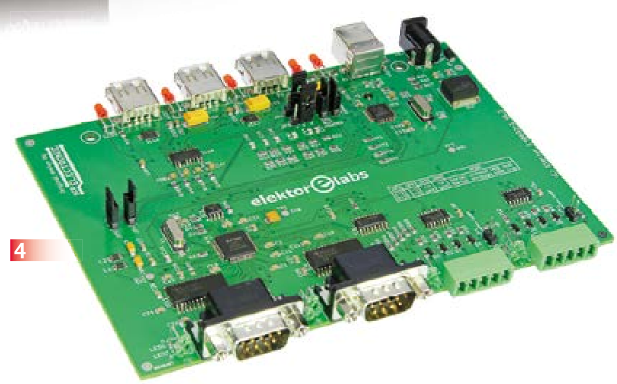
Remember that your document represents you when you're not there. Don't you want to be seen in the best possible light?

(140478)

1


2


3


4


The Authority on Magnetics and Inductors

1 Trilogy of Magnetics

A standard work in the hands of 15,000 design engineers, academics and researchers worldwide, Trilogy of Magnetics aims at familiarizing users with the characteristics and applications of inductive components. The Trilogy comprises: Basic Principles, Components, and Applications. Highlights include calculation, dimensioning and construction of customer-specific transformers, an introduction to frequency compensation, Class-D amplifier design and basic principles of Ethernet & Power-over-Ethernet. Furthermore Trilogy of Magnetics includes over 200 practical examples of filter circuits, audio & video circuits, interfaces, RF circuits, motor control units and SMPS. The book closes off with a keyword index and a summary of formulas.

728 pages • ISBN 978-3-89929-157-5
£42.95 • € 49.00 • US $67.00

The RPi in Control Applications

2 Raspberry Pi Hardware Projects

This book starts with an introduction to the Raspberry Pi computer and covers the topics of purchasing all the necessary equipment and installing/using the Linux operating system in command mode. Use of the user-friendly graphical desktop operating environment is explained using example applications. The RPi network interface is explained in simple steps and demonstrates how the computer can be accessed remotely from a desktop or a laptop computer. The remaining parts of the book cover the Python programming language, hardware development tools, hardware interface details, and RPi based hardware projects.

290 pages • ISBN 978-1-907920-29-5
£34.95 • € 39.95 • US $54.00

110 Elektor Editions, Over 2500 Articles

3 DVD Elektor 2000 through 2009

This DVD-ROM contains all circuits and projects published in Elektor magazine's year volumes 2000 through 2009. The 2500+ articles are ordered chronologically by release date (month/year), and arranged in alphabetical order. A global index allows you to search specific content across the whole DVD. Every article is printable using a simple print function. This DVD is packed with ideas, circuits and projects that are ideal for any electronics enthusiast, student or professional, regardless of whether they are at home or elsewhere.

ISBN 978-1-907920-28-8
£77.95 • € 89.00 • US $121.00

The e-engineer's delight

4 USB Hub feat. RS-232/RS-422/RS-485

This USB-to-serial converter allows you to connect two RS-232 and/or two RS-422/485 devices to your laptop or workstation through a standard USB (Universal Serial Bus) port. Thanks to the integrated USB hub the device does not 'consume' precious USB ports. Even better: It offers you two extra USB ports! The device is compatible with new and legacy serial devices, and is perfect for instrumentation and manufacturing applications.

Ready-built module Art.# 140033-91
£95.95 • € 110.00 • US $149.00

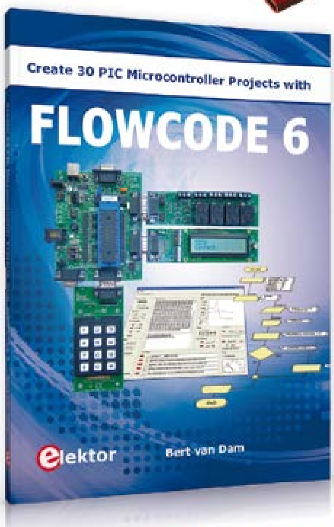
Fun to Build and Use Projects

5 Create 30 PIC Microcontroller Projects with Flowcode 6

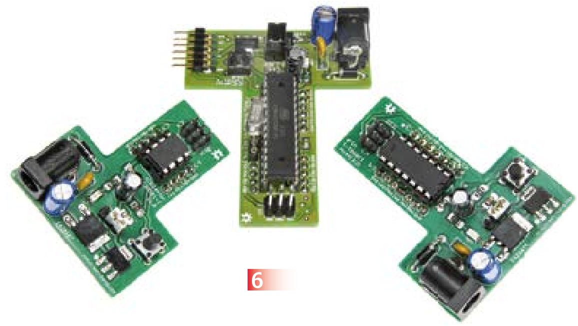
This book covers the use of Flowcode® version 6, a state-of-the-art, all-graphical based code development tool, for the purpose of developing PIC microcontroller applications at speed and with unprecedented ease. Without exception, the 30 projects in the book are fun to build and use. A secret doorbell, a youth deterrent, GPS tracking, persistence of vision (POV), and an Internet webserver are just a few examples of projects in the book waiting to be explored and mastered. This makes the publication

also available as E-book

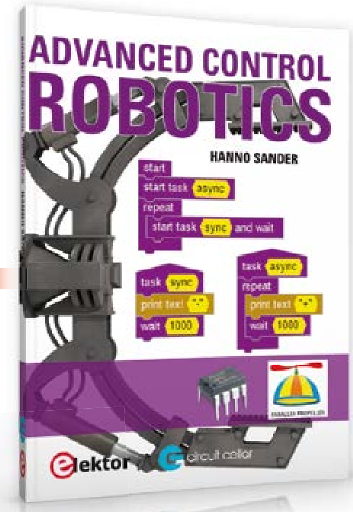
5



6



7



8

9

a perfect source of projects constantly challenging your hardware and software skills as you progress, resulting in advanced microcontroller applications you can be proud of. All sources referred in the book are available for free download, including the support software.

226 pages • ISBN 978-1-907920-30-1
£30.95 • € 34.95 • US $48.00

hree Sizes for Cheap & Fast AVR Prototyping

6 T-Boards

In response to the limitations posed by fixed-design, Arduino-style boards, Elektor has designed three mini-development boards that give developers more flexibility while still hosting the microcontroller and its supporting components. We're very excited to present the T-Boards! T-Boards are breadboard-friendly PCBs designed for simple and swift microcontroller prototyping using an Atmel ATmega328, ATtiny24-44-84 and ATtiny25-45-85 microcontrollers. Additionally, each T-Boards has an integrated 3.3V and 5V-selectable power supply, which assists in reducing the number of required jumper wires and allows for experimenting with lower power usage. Programming the microcontroller can be done in-circuit (ICSP).

Bundle of three boards
Art.# 130581-94
£33.95 • € 39.00 • US $53.00

Advanced Robot Technologies Made Easy

7 Advanced Control Robotics

It doesn't matter if you're building a line-following robot toy or tasked with designing a mobile system for an extraterrestrial exploratory mission: the more you know about advanced robotics technologies, the better you'll fare at your workbench. Hanno Sander's Advanced Control Robotics is intended to help roboticists of various skill levels take their designs to the next level with microcontrollers and the know-how to implement them effectively. Advanced Control Robotics simplifies the theory and best practices of advanced robot technologies. You're taught basic embedded design theory and presented handy code samples, essential schematics, and valuable design tips (from construction to debugging).

160 pages • ISBN 978-0-96301-333-0
£34.95 • € 39.95 • US $54.00

Build Your Own Robot
Measurement and Control using your PC

8 IO-Warrior Expension Board

Don't throw out your old PCs and notebooks or leave them gathering dust in the basement! They can be a useful resource: by adding this universal interface card an old PC can be pressed into service as a measurement and control hub. An IO-Warrior module on the I/F board takes care of USB communication, and source code is available that works with the free version of Visual Studio.

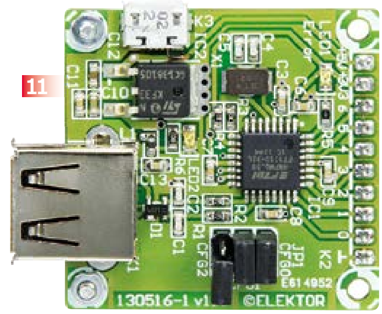
Ready-built IO-Warrior56 module
Art.# 130006-91
£34.95 • € 39.95 • $54.00

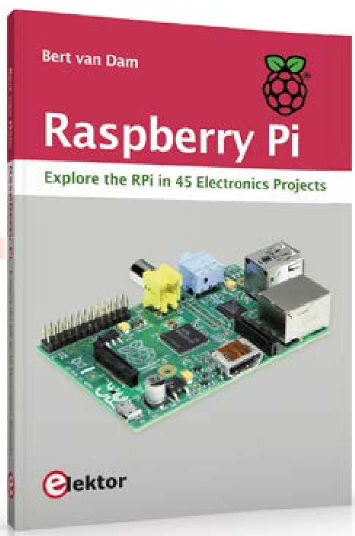
Nostalgia: The Fascination of 40 Years Ago

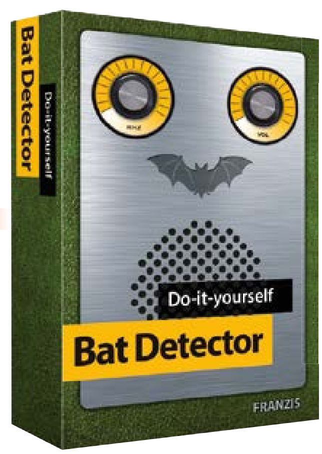
9 DIY VHF Retro Radio

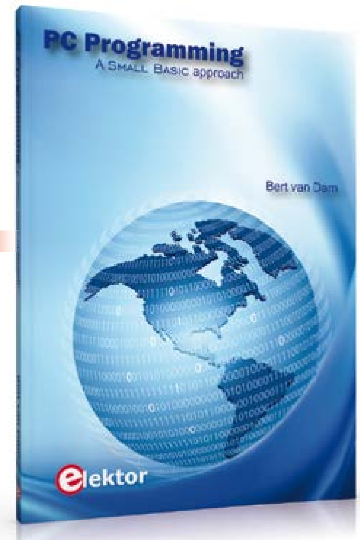
For anyone enthusiastic about Very High Frequency (VHF) radio, this all-in-one kit from our partner FRANZIS is ideal. This modern VHF radio in a stylish, retro body receives FM stations in the 87.5 MHz to 108 MHz band with good reception performance. You will mainly hear the powerful local stations in high sound quality. However, the sensitivity of the receiver also allows you to listen to remote stations at times. This radio assembly kit contains all required parts, including casing, printed circuit board, rod antenna and all necessary components and can be assembled easily and enjoyably. High quality components, high sensitivity and a rod antenna provide the best radio reception possible.

All-in-one kit
Art.# 140260-91
£26.95 • € 29.95 • US $41.00









Explore the RPi in 45 Electronics Projects

10 Raspberry Pi

This book addresses one of the strongest aspects of the Raspberry Pi: the ability to combine hands-on electronics and programming. No fewer than 45 exciting and compelling projects are discussed and elaborated in detail. From a flashing lights to driving an electromotor; from processing and generating analog signals to a lux meter and a temperature control. We also move to more complex projects like a motor speed controller, a webserver with CGI, client-server applications and Xwindows programs. Each project has details of the way it got designed that way. The process of reading, building and programming not only provides insight into the Raspberry Pi, Python, and the electronic parts used, but also enables you to modify or extend the projects any way you like.

288 pages • ISBN 978-1-907920-27-1
£34.95 • € 39.95 • US $56.40

Android User Interface Builder

11 Android Breakout Board

The FTDI FT311D is a flexible bridge that can interface your circuit to an Android smartphone or tablet. This Elektor Android Breakout Board offers options for seven digital outputs, four PWM outputs, asynchronous serial and I2C and SPI interfaces. The board is compatible with Android 3.1 (Honeycomb) or higher (Android Open Accessory Mode should be supported).

Ready-built module
Art.# 130516-91
£26.95 • € 29.95 • US $41.00

12 Do-it-yourself Bat Detector

Who says you can't hear bats? With this proper gizmo, you can–and much more! Build your own high-quality ultrasonic bat detector and experience these sounds for yourself! This kit for home assembly makes it an easy and fool-proof process. The printed circuit board comes with many SMD components premounted. All you have to do is solder in a few parts and connect the circuit board to the microphone, the loudspeaker and the controls. State of the art integrated circuits ensure high sensitivity and volume. Once you have assembled the bat detector, you can start exploring right away!

All-in-one-kit
Art.# 140259-91
£26.95 • € 29.95 • US $41.00

A Small Basic approach

13 PC Programming

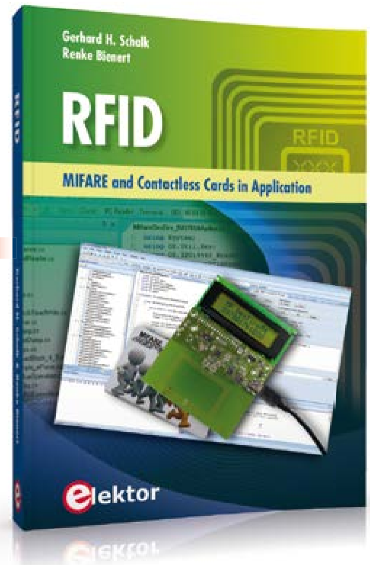
There are many different PC programming languages available on the market. They all assume that you have, or want to have, a knack for technology and difficult to read commands. In this book we take a practical approach to programming. We assume that you simply want to write a PC program, and write it quickly. Not in a professional environment, not in order to start a new career, but for plain and simple fun... or just to get a task done. Therefore we use Small Basic. You will have an application up and running in a matter of minutes. You will understand exactly how it works and be able to write text programs, graphical user interfaces, and advanced drivers.

194 pages • ISBN 978-1-1-907920-26-4
£30.95 • € 34.50 • US $47.00

MIFARE and Contactless Cards in Application

14 RFID

MIFARE is the most widely used RFID technology, and this book provides a practical and comprehensive introduction to it. Among other things, the initial chapters cover physical fundamentals, relevant standards, RFID

14


15


16


antenna design, security considerations and cryptography. The complete design of a reader's hardware and software is described in detail. The reader's firmware and the associated PC software support programming using any .NET language. The specially developed PC program, "Smart Card Magic.NET", is a simple development environment that supports sending commands to a card at the click of a mouse, as well as the ability to create C# scripts. Alternatively, one may follow all of the examples using Visual Studio 2010 Express Edition. Finally, the major smart card reader API standards are introduced. The focus is on programming contactless smartcards using standard PC/SC readers using C/C++, Java and C#.

484 pages • ISBN 978-1-907920-14-1
£43.95 • € 49.90 • US $68.00

15 Arduino Extension Shield

This shield intended to augment the Arduino Uno offers starters a text display, LEDs and pushbuttons to provide a good basis for their projects. For experienced users two extension connectors are provided which can be used to connect relay modules, wireless modules and many other devices. This shield thus ameliorates the Arduino Uno's most significant shortcomings on on-board peripherals.

Ready-built module
Art.# 140009-91
£23.95 • € 26.95 • US $37.00

Ideal reading for students and engineers

16 Practical Digital Signal Processing using Microcontrollers

This book on Digital Signal Processing (DSP) reflects the growing importance of discrete time signals and their use in everyday microcontroller based systems. The author presents the basic theory of DSP with minimum mathematical treatment and teaches the reader how to design and implement DSP algorithms using popular PIC microcontrollers. The author's approach is practical and the book is backed with many worked examples and tested and working microcontroller programs. The book should be ideal reading for students at all levels and for the practicing engineers who may want to design and develop intelligent DSP based systems. Undergraduate students should find the theory and the practical projects invaluable during their final year projects. Similarly, postgraduate students should be able to develop advanced DSP based projects with the aid of the book.

428 pages • ISBN 978-1-907920-21-9
£43.95 • € 49.90 • US $68.00

Further Information and Ordering: www.elektor.com/store
or contact customer service for your region

UK / ROW
Elektor International Media
78 York Street
London - W1H 1DP United Kingdom
Phone: +44 20 7692 8344
E-mail: service@elektor.com

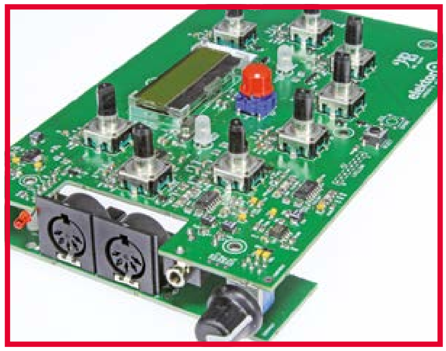
USA / CANADA
Elektor US
111 Founders Plaza, Suite 300
East Hartford, CT 06108 USA
Phone: 860.289.0800
E-mail: service@elektor.com

UPCOMING IN ELEKTOR

January & February 2015 Double Edition

Traditionally we start the new year with a Jumbo-size double edition. On the production schedule are a ton of projects, ideas and tips. The projects in particular range from complex and ambitious down to single-transistor designs you should be able to build in an afternoon. The mix consists of microcontroller projects, measurement & control applications, analog electronics and small experimental stuff. In addition to the headliners pictured here you can look forward to:

- Simple FM Receiver
- LED DC/DC Booster
- Humidity Sensor eBoB
- True-RMS Converter for DMM
- CMOS IR Modulator
- XBee T-Board
- Timecode Visualization Clock
- Multitester with ELPP
- Retronics XXL



J:B Synthesizer

After years of silence, Elektor has a music synthesizer project on its pages. This time it's not an analog monster the size of a cupboard but a compact digital box not much larger than a fat novel. Based on the J²B ARM Cortex-M3 board from September 2011 but reworked down to the processor, this synth is guaranteed to give hours of fun playing and experimenting. The project is totally open source and open-hardware, and has a lot of computing power in store.

Platino Signal Generator

Our line of Platino based test & measurement equipment is extended with a signal generator with sinewave, square wave, sawtooth and pulse output signals. Users can also programs their own waveforms, and the instrument has an input for frequency modulation. Operation is straightforward and lucid thanks to a rotary encoder and a 4-line LCD.

Edition: 1/2015, no. 457/458, January & February. Publication date: January 8, 2015. Content and article titles subject to change.